António Pedro Ferreira Silva

Licenciado em Engenharia Informática

Uma Abordagem Ágil para Transformar Modelos Cognitivos em Modelos Comportamentais e de Domínio

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática

Orientador :	João Araújo, Dr. Prof, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa
Co-orientador :	Fernando Wanderley, Doutorando, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa

Júri:

Presidente:	Dr. Prof Francisco de Moura e Castro Ascenção de Azevedo
Arguentes:	Dr. Prof Ademar Manuel Teixeira de Aguiar
	Dr. Prof João Baptista da Silva Araújo Júnior

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2014

Uma Abordagem Ágil para Transformar Modelos Cognitivos em Modelos Comportamentais e de Domínio

Aos meus pais, que me tornaram na pessoa que sou hoje.

Agradecimentos

Gostaria de agradecer todo o apoio dos meus orientadores, Prof. João Araújo e Fernando Wanderley, por me guiarem por todas as dúvidas e obstáculos que ocorreram durante a realização desta dissertação. Muito obrigado pela oportunidade, disponibilidade, reuniões, avisos e conselhos que me fizeram crescer e tornaram possível a realização deste projeto. Gostaria de agradecer ao Fernando, por toda a motivação e atenção que partilhou e pelo trabalho realizado durante este ano no laboratório da faculdade que criaram laços de amizade.

A todos os meus amigos que me apoiaram durante esta jornada de conclusão de curso, que me ajudaram e motivaram para concluir esta etapa. A todos os amigos que encontrei nesta passagem pela universidade.

Finalmente, à minha família porque sem ela nada disto seria possível, um muito obrigado por tudo o que fizeram para que conseguisse realizar esta tarefa.

Muito obrigado a todos.

Resumo

No processo de desenvolvimento de *software*, um dos problemas recorrentes é garantir que as expectativas das partes interessadas (*stakeholders*) estão a ser satisfeitas. Expectativas essas que correspondem ao comportamento do sistema. A disciplina de Engenharia de Requisitos estuda a melhor forma de capturar, especificar, validar e gerir requisitos. No entanto, os modelos utilizados que expressam os comportamentos, muitas das vezes, não são entendidos por parte dos *stakeholders*.

Com a introdução das metodologias ágeis, que se baseiam no princípio de uma colaboração ativa e contínua dos *stakeholders* durante o desenvolvimento de *software*, os pressupostos da disciplina de Engenharia de Requisitos foram questionados e fizeram nascer novas práticas. Uma prática que emergiu foi o Desenvolvimento Orientado ao Comportamento (BDD). Surgiu com a finalidade de dar a capacidade aos *stakeholders* de expressarem, sob a forma textual, o comportamento que desejam para o seu *software*, de forma simples e sucinta. No entanto, não existindo restrições nem validações automáticas na escrita dos *stakeholders*, é criada a possibilidade de introdução de ambiguidade e perda de informação quando a equipa de desenvolvimento utiliza os cenários descritos.

Dado este problema, propomos uma abordagem em que os *stakeholders* consigam especificar cenários comportamentais, de forma cognitiva, com uma representação simples, clara e mais precisa. Criamos duas linguagens, o *DomainMap* e o *BehaviorMap*, que estão relacionadas entre si: uma para representar modelos de domínio e outra para representar cenários BDD, respetivamente. Foi executada uma experiência com 15 indivíduos para comparar o esforço cognitivo na compreensão entre cenários *BehaviorMap* e cenários textuais e os resultados mostraram que o *BehaviorMap* teve melhor desempenho.

Palavras-chave: Engenharia de Requisitos, Desenvolvimento Orientado ao Comportamento, Modelação Ágil, Testes de Aceitação

Abstract

In the software development process, one of the recurring problems is to ensure that the expectations of stakeholders are being met. These expectations must match the system's behavior. The Requirements Engineering discipline studies how to capture, specify, validate and manage requirements. However, the models used to express behaviors are not usually understood by the stakeholders.

With the introduction of the Agile methodologies, which have an ongoing collaboration of stakeholders during the development of software, the assumptions of the discipline of requirements engineering were questioned and gave rise to new practices. One of them was *Behavior-Driven Development*, emerged with the purpose of giving stakeholders the ability to express, in the textual form, the behavior they desire for their software. Nevertheless, there are no restrictions nor validations when the behaviour is written by them. That may introduce subjectivity and loss of information when the development team reads the behaviors described.

Given this problem, the ideal solution would be to have an approach where stakeholders were able to specify the desired behavior, in a cognitive and creative way, with a simple and clear representation, ensuring the understanding of all the involved. We created two languages, the DomainMap language to represent domain models and the BehaviorMap language to represent BDD scenarios. An experiment was performed with 15 subjects to compare the cognitive effort when reading and understanding textual and BehaviorMap scenarios. The results showed that the BehaviorMap scenarios had an average lowest cognitive effort.

Keywords: Requirements Engineering, Behavior-Driven Development, Agile Modeling, Acceptance Testing

Lista de Acrónimos

MDD - *Model-Driven Development*

MDE - *Model-Driven Engineering*

XP - *eXtreme Programming*

DSL - *Domain Specific Language*

BDD - *Behavior-Driven Development*

TDD - *Test-Driven Development*

DDD - *Domain-Driven Design*

ATDD - *Acceptance-test Driven Development*

CSS - *Cascading Style Sheets*

UML - *Unified Modeling Language*

OCL - *Object Constraint Language*

EVL - *Epsilon Validation Language*

EOL - *Epsilon Object Language*

GMF - *Graphical Modeling Framework*

Conteúdo

Lista de Figuras

Lista de Tabelas

Listagens

1

Introdução

1.1 Contexto

No processo de desenvolvimento de *software*, um dos problemas recorrentes é garantir que as expectativas das partes interessadas (*stakeholders*[1]) estão a ser satisfeitas [Bro87]. Expectativas essas que correspondem ao comportamento do sistema. De acordo com [Gro14], requisitos ambíguos e mal compreendidos são a maior causa para o insucesso de um projeto.

Com o objetivo de representar estas expectativas e os objetivos dos *stakeholders*, a disciplina de engenharia de requisitos (ER) dedica-se a estudar a melhor forma de como capturar, especificar, validar e gerir requisitos [KS98]. Nessas atividades são produzidos modelos (gráficos ou textuais) para expressarem os desejos dos *stakeholders*. Porém, as suas representações e notações são, muitas das vezes, entendidas apenas por especialistas de TI[2] [Cai+13]. As representações e notações utilizadas nos modelos de ER não têm em consideração a opinião dos *stakeholders* [Cai+13]. Tendo em conta que o custo monetário e temporal de encontrar erros de *design* aumenta drasticamente ao longo do desenvolvimento de *software*, é essencial que os requisitos capturados pela equipa de desenvolvimento sejam validados por parte dos *stakeholders* o mais cedo possível. No entanto, essa tarefa pode complicar-se caso os *stakeholders* não compreendam as notações utilizadas no modelo de requisitos utilizados.

Com a introdução de metodologias ágeis, que se estão a tornar populares por equipas de desenvolvimento [Wes13], os pressupostos e técnicas criadas pela disciplina de ER

[1] Nesta dissertação *stakeholders* referem-se a clientes, utilizadores finais, etc e não a membros da equipa de desenvolvimento

[2] TI - Tecnologias de Informação

foram questionados. Contrariando um processo longo e detalhado do modelo Cascata[3] e a não explicitação ou omissão de *stakeholders* ao longo do processo, os princípios das metodologias ágeis salientam que a presença dos *stakeholders* e curtos ciclos de desenvolvimento são essenciais para confirmar e garantir que o processo vai ao encontro das suas expectativas.

Dentro das metodologias ágeis, uma das práticas de captura e especificação de requisitos é o Desenvolvimento Orientado ao Comportamento, em inglês (*Behavior-Driven Development*- BDD) [SW11]. O BDD surgiu com a finalidade de dar a capacidade aos *stakeholders* de expressarem, com a linguagem de domínio a que pertencem, o comportamento que desejam para o seu *software*. Os comportamentos expressos dão origem a testes de aceitação e guiam a equipa de desenvolvimento às necessidades dos *stakeholders*.

1.2 Motivação

Até ao momento, as ferramentas disponíveis de suporte ao BDD permitem apenas expressar os comportamentos através de linguagens textuais de uma forma livre [4,5,6]. Essa escrita livre pode causar interpretações erróneas dos cenários por parte da equipa de desenvolvimento e conduzir a casos de testes discordantes das expectativas dos *stakeholders*. Consequentemente, essa má interpretação resulta num esforço de trabalho inglório por parte da equipa de desenvolvimento, num erro que poder ser diminuído desde as fases inicias do desenvolvimento [BA04].

Dado este problema, o ideal seria ter uma abordagem em que os *stakeholders* conseguissem especificar o comportamento desejado, de forma cognitiva e criativa, com uma representação simples e clara, assegurando a compreensão de todos os envolvidos. Um desafio interessante seria a possibilidade de construir um modelo cognitivo capaz de ser compreendido por **todos** os *stakeholders* envolvidos, e que a informação capturada nesse mesmo modelo não fosse perdida no processo de desenvolvimento. Adicionalmente, utilizar essa informação para gerar casos de testes automaticamente e com uma maior cobertura, em comparação com as ferramentas atuais, reforçando assim a garantia da satisfação das necessidades dos *stakeholders*.

1.3 Objetivo da dissertação

Tirando proveito das Linguagens Especificas de Domínio, em inglês (*Domain Specific Languages*- DSL) e recorrendo a técnicas de transformações, desenvolvemos duas linguagens gráficas, com base em mapas mentais, visando a compreensão entre as duas partes:

[3] Modelo Cascata - http://en.wikipedia.org/wiki/Waterfall_model
[4] *JBehave* - http://jbehave.org
[5] *Cocumber* - http://cukes.info
[6] *SpecFlow* - http://www.specflow.org

stakeholders e equipa de desenvolvimento. Um mapa mental é um modelo cognitivo utilizado para representar conhecimento [BB93]. As linguagens desenvolvidas representam modelos de domínio, com a finalidade de agrupar a informação do domínio em questão, e cenários comportamentais BDD que permitem a execução de transformações automáticas para testes de aceitação na plataforma *JUnit*. Recorrendo ao Desenvolvimento Orientado a Modelos, em inglês (*Model-Driven Development*- MDD), um processo de desenvolvimento de *software* que se centra na construção e transformação de modelos para guiar o desenvolvimento do sistema, criámos um modelo para representar modelos de domínio, cenários BDD e transformações para testes de aceitação.

O objetivo desta dissertação é especificar e implementar um *framework*, *SnapMind*, que consiste em duas DSLs com base em mapas mentais para representar modelos de domínio e melhorar a representação de cenários BDD. Essas linguagens são inseridas no processo de desenvolvimento do BDD de forma a facilitar a colaboração dos *stakeholders* no processo de captura e validação de requisitos. A linguagem de modelação de modelos de domínio tem o propósito de agrupar informação relevante do mesmo para facilitar a sua compreensão. Porém, essa informação é também utilizada na construção de cenários BDD. A linguagem dos cenários comportamentais tem a finalidade de especificar cenários BDD. Possui a capacidade de aceitar informação proveniente dum modelo de domínio, para facilitar a escrita dos cenários, e a transformação automática de cenários para testes de aceitação.

1.4 Principais contribuições

Ao realizar esta dissertação, esperamos contribuir com duas DSLs para melhorar a utilização do BDD. A primeira DSL representa modelos de domínio, de forma a representar o domínio da aplicação a desenvolver e consequentemente integrar essa mesma informação na construção de cenários BDD. A segunda DSL recai sobre cenários BDD, onde queremos contribuir com uma melhoria na escrita dos mesmos, impondo restrições na mesma e validando o seu conteúdo através da incorporação da informação proveniente dum modelo de domínio. Adicionalmente, queremos contribuir com a geração automática de testes de aceitação e aumento da sua cobertura. Por último, ambas as linguagens desenvolvidas incorporam características de modelos cognitivos, nomeadamente mapas mentais, visando melhorar a compreensão por parte dos *stakeholders*.

Parte das contribuições desta dissertação foram publicadas no IEEE 4th International Model-Driven Requirements Engineering Workshop, com o artigo intitulado"SnapMind: A framework to support consistency and validation of model-based requirements in agile development" [Wan+14], apresentado em Karlskrona, Suécia, em 25 de Agosto de 2014.

1.5 Organização do documento

Os restantes capítulos pertencentes a esta dissertação vão abordar os seguintes temas:

Capítulo 2 - Modelação ágil e Engenharia de Requisitos: a definição de Modelação Ágil e Engenharia de Requisitos, como também de *User Stories* e algumas metodologias ágeis praticadas atualmente são os temas abordados neste capítulo.
Capítulo 3 - Desenvolvimento baseado em cenários: abordar a evolução desde o Desenvolvimento Orientado a Testes até à evolução do BDD.
Capítulo 4 - Desenvolvimento Orientado a Modelos e Linguagens Específicas de Domínio: a introdução de DSL e de MDD serão os temas abordados neste capítulo. Irá também cobrir ferramentas de desenvolvimento de DSL.
Capítulo 5 - Trabalho relacionado: expor o resultado de uma pesquisa com intuito de investigar quais são os modelos ágeis utilizados para representar comportamento de *software*, a adoção do BDD em processos ágeis e abordagens de requisitos centrados em utilizadores.
Capítulo 6 - Abordagem: apresentação do processo desenvolvido proposto e das linguagens desenvolvidas, *DomainMap* e *BehaviorMap*.
Capítulo 7 - Avaliação: expor as avaliações realizadas e os seus resultados para a avaliação da linguagem *BehaviorMap*.
Capítulo 8 - Conclusão: resumo das contribuições e trabalho futuro provenientes da dissertação.

2

Modelação Ágil e Engenharia de Requisitos

2.1 Modelação Ágil

A Modelação Ágil, em inglês (*Agile Modeling*- AM), é uma atividade da metodologia ágil que engloba um conjunto de boas práticas e valores. Tem como alvo melhorar a eficácia de comunicação dos modelos produzidos, pretendendo integrar *stakeholders* no processo de desenvolvimento. Esta atividade tem três objetivos [Amb02]: **(1)** definir e demonstrar como colocar em prática os valores e princípios para uma modelação leve e eficaz; **(2)** evitar comportamentos centrados em código e utilização de *sketches* para delinear pensamentos; **(3)** estudar a forma de melhorar as técnicas de modelação usadas por processos tais como o RUP[1] e EUP[2]. Os cinco valores da génese do AM são: a comunicação, simplicidade, *feedback*, coragem e humildade.

2.1.1 Modelação Ágil e a importância da comunicação

A preocupação de uma integração ativa e frequente dos *stakeholders* no desenvolvimento do *software* resultou em alterações na disciplina de Engenharia de Requisitos [PEM03; LQ10]. Surgiram novas práticas que visavam os valores do AM e permitiam uma fácil integração dos *stakeholders*. O foco passou a ser a procura de uma comunicação eficaz entre todos os *stakeholders*, i.e., instigar a clareza na transmissão dos requisitos à equipa de desenvolvimento sem perda de informação. Esta comunicação eficaz resulta numa maior

[1] RUP - Rational Unified Process: *framework* para desenvolver software e gestão de projetos
[2] EUP - Enterprise Unified Process - Extensão do RUP

probabilidade de sucesso de realização do projeto [Amb02]. *Alistair Cockburn* [Coc02] descreveu diversas formas de comunicação entre indivíduos. Os resultados da sua investigação podem ser visualizados na Figura 2.1 (Adotada por *Scott Ambler* [AC14]). Ao observar o gráfico, pode-se concluir que a interação melhora com a proximidade dos participantes e o uso de uma ferramenta simples nesse canal de comunicação leva a uma grande eficácia da mesma. As ferramentas, regra geral, devem ser as mais simples possível, sem requerer treino para a sua utilização. Relativamente à escolha da forma de comunicação, *Cockburn* concluiu também que as preferências e gostos dos interlocutores têm de ser tomadas em conta para que não seja criada nenhuma ansiedade de ambas as partes.

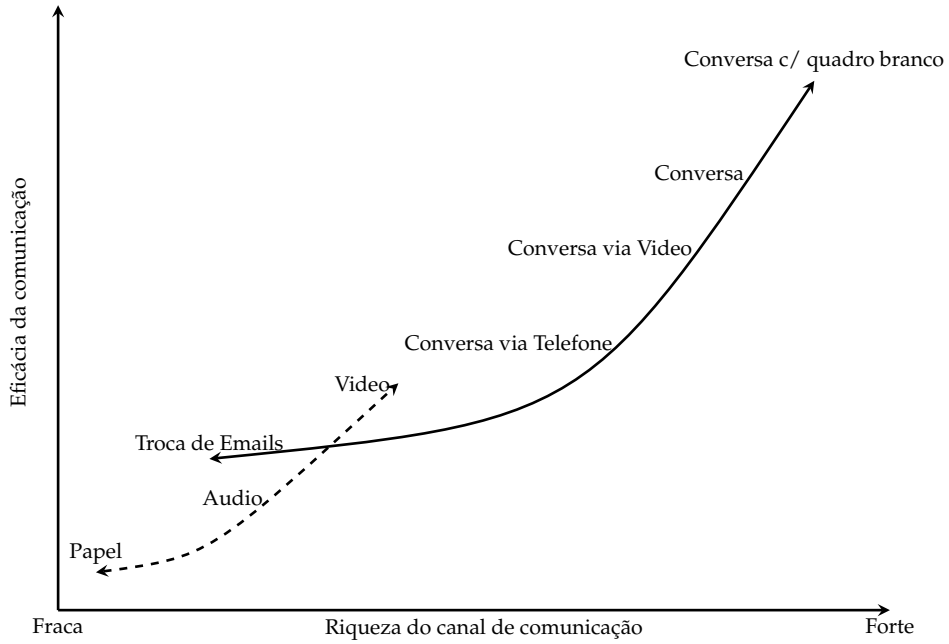


Figura 2.1: Gráfico da eficácia da comunicação relacionada com o canal de comunicação [AC14]

A implicação deste estudo na modelação ágil resultou num grande esforço por parte das equipas de desenvolvimento a procurarem uma comunicação eficaz com os *stakeholders* [Amb14]. Esta é adaptada continuamente para garantir entendimento de todos ao longo do processo de desenvolvimento. O principal fator para produzir uma comunicação eficaz é ter mecanismos de inclusão de todos os *stakeholders* e fazer com que todos se sintam confortáveis com a sua participação [Coc02].

2.1.2 Metodologias Ágeis

SCRUM

O SCRUM é uma metodologia de desenvolvimento ágil que tem ganho imensa popularidade [Wes13]. Este é distinguido pelas seguintes características: **(1)** o trabalho é agrupado em *sprints* com 30 dias de duração, onde resultam três *releases* durante cada *sprint*;

(2) todo o trabalho desenvolvido é registado no *product backlog*, que inclui os requisitos (novos e antigos) a desenvolver, os defeitos registados e as atividades de *design*; **(3)** a equipa é dirigida por um *Scrum Master* que corresponde a um líder de equipa e comunica com o cliente através de um *product owner*; **(4)** diariamente são conduzidas reuniões curtas, onde os participantes estão todos de pé a descrever o que fizeram e o que resta fazer. A Figura 2.2 exemplifica o processo do SCRUM.

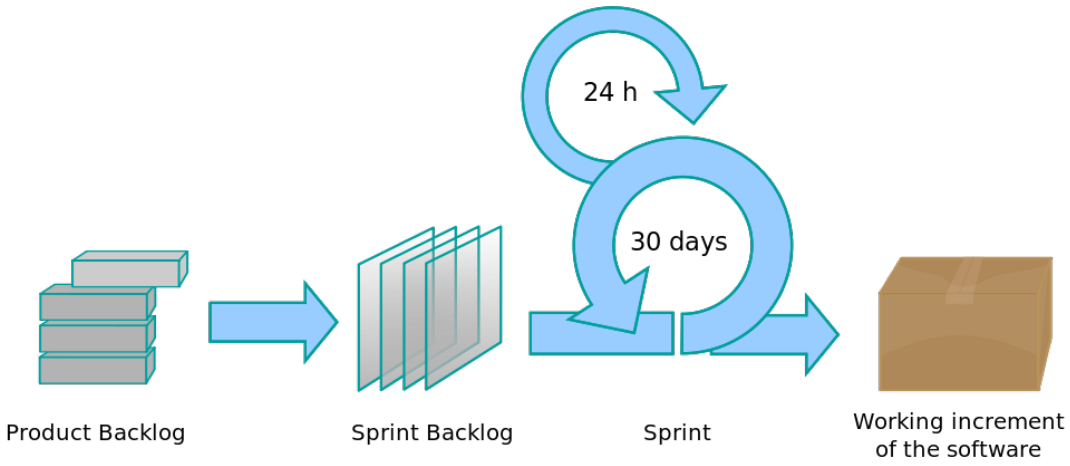


Figura 2.2: Processo de desenvolvimento SCRUM [Sch04]

eXtreme Programming

O *eXtreme Programming* (XP) é uma metodologia de desenvolvimento de *software* que foi desenhada com a finalidade de ter uma boa resposta face a requisitos vagos e a mudanças dos mesmos por parte do cliente. Pensado para equipas pequenas e partilhando os princípios de uma metodologia ágil, tem ciclos de desenvolvimentos curtos, com o intuito de *feedback* constante por parte do cliente, onde o mesmo integra a equipa de desenvolvimento [BA04].

Ao contrário dos modelos tradicionais (e.g. desenvolvimento em cascata), o XP tem como princípio a construção de um sistema minimalista o mais cedo possível, com um *design* simples a ser evoluído ao longo do tempo. Desde o início, são feitos testes unitários antes da produção de código e são executados inúmeros testes do sistema ao longo do desenvolvimento. Outra característica presente no XP é o facto de as equipas de desenvolvimento serem pequenas, onde os programadores trabalham em pares: um produz código e outro analisa, trocando regularmente.

2.1.3 Desafios das Metologias Ágeis

As metodologias ágeis são relativamente recentes no âmbito de desenvolvimento de *software*. São apresentadas como sendo uma revolução inovadora e eficaz. A liberdade de

evitar planos detalhados, focando o esforço em pessoas e nas suas interações permanece nos pilares do desenvolvimento ágil [Bec+14]. Assim os projetos com equipas ágeis apresentam elementos característicos como: equipas pequenas, *stakholders* envolvidos, planeamento continuo, equipas multifuncionais, organizações que não separam equipas até que cada membro seja competente em métodos ágeis [laz+08]. Estas características são comuns para desenvolver projetos de média e pequena complexidade. No caso de projetos de larga complexidade e escala, o seu favoritismo baixa [Bar+10].

A liberdade requer muita disciplina para conduzir uma equipa de desenvolvimento num caminho correto para satisfazer as necessidades dos *stakeholders* [laz+08]. Em projetos de grande dimensão existe uma dificuldade em coordenar várias pequenas equipas de forma a garantir que todas cumprem os princípios ágeis. Práticas ágeis, como o *refactoring*, não conseguem ser facilmente escaladas entre equipas de forma a não existirem divergências no produto. A introdução de alterações de requisitos em diversos ciclos de desenvolvimento, carecendo dum estudo detalhado do seu impacto, pode resultar em custos monetários e temporais superiores em relação a uma abordagem planeada. Outro impacto possível dum estudo carente de requisitos é uma arquitetura não respeitante de requisitos não funcionais, o que pode comprometer todo o produto (ou serviço) a ser desenvolvido.

Como uma forma de combater este problema, têm sido estudadas metodologias híbridas, i.e., a composição de ideias ágeis com processos melhor definidos, de forma a organizar a liberdade inerente. Existem trabalhos que relatam a incorporação dessas abordagens em empresas (*Yahoo*, *CapitalOne*, *BMC Software*) para tentar incorporar processos híbridos no seu processo de desenvolvimento [laz+08].

Esta dissertação foca-se na tentativa de melhorar o processo BDD e a comunicação entre a equipa de desenvolvimento e *stakeholders*, ignorando questões de planeamento de projetos ágeis mencionados anteriormente. Propomos uma abordagem com base em práticas ágeis, no entanto, não existe nenhum impedimento em inserir as linguagens definidas noutras metodologias.

2.2 Engenharia de Requisitos

A disciplina de Engenharia de Requisitos, em inglês (*Requirements Engineering*- RE), é uma vertente da Engenharia de *software* que estuda a forma de descrever o comportamento de um sistema a ser desenvolvido [Som10]. O comportamento do sistema pode ser considerado em serviços que representam as necessidades dos *stakeholders* e as restrições das operações do sistema. Os objetivos da disciplina de RE são: capturar, analisar, verificar e documentar todos os serviços e restrições do sistema a ser desenvolvido.

Os requisitos podem ser distintos em duas categorias: funcionais e não funcionais. Os **requisitos funcionais** descrevem o que o sistema deve fazer, o seu comportamento em certas situações. São esses requisitos que expressam as necessidades dos *stakeholders*. São normalmente escritos de forma abstrata para que os *stakeholders* consigam participar

na sua construção e análise. Caso o requisito funcional seja de uma vertente do sistema, esse é escrito com mais detalhe.

Os **requisitos não funcionais** expressam características do sistema como um todo. São ortogonais a todo o sistema e representam características como, por exemplo: fiabilidade, segurança, tempo de resposta. Estes requisitos implicam restrições na construção do sistema, na sua implementação e na forma como os vários constituintes se relacionam e comportam. São normalmente considerados prioritários aos requisitos funcionais porque afectam o sistema com um todo e, caso sejam negligenciados, tornam-o obsoleto.

2.2.1 Abordagens de Engenharia de Requisitos

Existem diversas abordagens de descrição de requisitos na disciplina de engenharia de requisitos. Entre elas destacamos:

1. **Engenharia de Requisitos Orientada a Objetos:** esta abordagem de desenvolvimento de *software* tem a característica de construir o sistema que consiste na interação de objetos. Objetos esses que representam os requisitos do *software*. Cada objeto é constituído por atributos e operações e relaciona-se com outros objetos. Todos os objetos de um sistema devem representar todos os requisitos que o sistema deve obedecer [Boo94];

2. **Engenharia de Requisitos Orientada a Aspectos:** é uma abordagem caracterizada pela aplicação de uma modularização avançada ao nível de requisitos. Fá-lo identificando aspectos (propriedades transversais) entre vários requisitos. Este tipo de desenvolvimento leva a implementações desses aspectos em diferentes módulos que, por sua vez, são instanciados automaticamente quando requisitados [Ara+05];

3. **Engenharia de Requisitos Orientada a Objetivos:** caracteriza-se pela identificação dos requisitos de *software* a partir de objetivos. Um objetivo é algo que o sistema a ser desenvolvido deve alcançar, seja funcional ou não funcional. Este processo agrega vários objetivos em diferentes níveis de abstração, permitindo a decomposição de objetivos grandes em requisitos [Van01];

4. **Engenharia de Requisitos Orientada a *Viewpoints*:** nesta abordagem, os requisitos são estruturados de acordo com pontos de vista associados à origem do requisito [Kot+96];

5. **Engenharia de Requisitos Orientada a Cenários:** esta prática envolve trabalhar diretamente com os *stakeholders* com o intuito de identificar cenários de funcionamento do sistema a desenvolver. São escritos em texto, com linguagem dos *stakeholders* e podem incluir diagramas [Som10]. Na secção seguinte expomos alguns dos possíveis modelos comportamentais que podem ser utilizados em conjunto com os cenários.

2.2.2 Modelos Comportamentais

Os modelos comportamentais são modelos de desenho que têm o papel de expressarem o comportamento dinâmico do sistema no decorrer da sua execução [Som10]. Demonstram como o sistema se deve comportar, dado um certo conjunto de estímulos vindos do ambiente inserido. Os estímulos podem ser considerados como dados, e.g. *inputs* ou eventos, como chamadas de funções ou como um *trigger*.

O UML[3] possui vários modelos comportamentais próprios para modelação de sistemas. Ei-los, com uma pequena definição [RJB05]:

1. **Diagramas de Atividade:** servem para modelar aspectos sequenciais ou concorrentes de um sistema de diversas atividades. Uma atividade é a consequência do sinal de *input* dado a uma máquina de estados (mudança de estado e *output*). Este diagrama tem a capacidade de visualização, especificação, construção e documentação do comportamento dinâmico de um grupo de objetos relacionados pelas suas atividades;
2. **Diagramas de Sequência:** servem para representar comportamentos de objetos numa situação particular, oferecendo uma visão temporal. A cada objeto é associada uma linha temporal, onde são inseridas as suas interações com outros objetos por via de mensagens;
3. **Diagramas de Estados:** são diagramas utilizados para especificar a linha de vida de uma instância de uma classe, um *Use Case* ou de um sistema com a utilização de estados. Um estado de um objeto é uma condição ou situação durante o tempo de vida de um objeto em que o mesmo satisfaz uma condição, executa uma atividade e espera por um evento. Os diagramas de estado mapeiam os diversos estados que o componente possui e também os sinais que representam as transições possíveis;
4. **Diagramas de Colaboração:** são construídos para delinear a estrutura de vários objetos numa determinada interação. Os objetos são inseridos como vértices de um grafo. Após isso, acrescentam-se arcos de ligação entre eles. Por fim, são adicionadas mensagens simbólicas aos arcos com informação relevante da interação;
5. **Diagramas de *Use Case*:** têm o objetivo de modelar o contexto do sistema, dos subsistemas, as suas relações e os requisitos que cada um representa. São necessários para ter uma visão geral do sistema a desenvolver, porque consegue-se visualizar, especificar e documentar como os elementos são utilizados num certo contexto;

2.3 Modelos ágeis e requisitos

Num processo ágil, a construção de um sistema é feita incrementalmente e iterativamente, obtendo *feedback* de validação por parte dos *stakeholders* no final de cada ciclo

[3]UML - Linguagem de modelação orientado aos objetos criada por Grady Booch.

iterativo. Isso resultou no nascimento de práticas focadas em capturar pequenas expectativas dos *stakeholders*. Essas práticas de ER são pouco explícitas [Coc02] e são utilizadas dependendo da situação do desenvolvimento do projeto [CR08]. No entanto, existe uma que é utilizada em várias metodologias ágeis (e.g. SCRUM e XP) denominada *User Stories*, que são cenários em forma textual que especificam a interação do utilizador com o sistema.

2.3.1 User Stories

As *User Stories* são a prática mais popular de captura de requisitos nas metodologias ágeis [MA04; Lef11]. Originalmente utilizadas no contexto do *eXtreme Programming*, têm vindo a ser utilizadas desde então. As *User Stories* são pequenos cenários do sistema para ajudar os *stakeholders* a partilhar comportamentos esperados do sistema a desenvolver. Visam obter qual o requisito a implementar, o porquê da sua existência e qual o *business value* (valor de negócio) associado à sua implementação. São escritas com uma linguagem próxima do entendimento dos *stakeholders*, visando uma comunicação eficaz com a equipa de desenvolvimento. Servem como guia para a equipa de desenvolvimento, guiando futuras conversas e gerando discussões [MA04].

Cada *User Story*, ao ser criada, tem de ter presente quatro propriedades: **(1)** tem de ser descritiva, i.e., ser possível extrair uma expectativa do *stakeholder* ou uma funcionalidade do sistema; **(2)** tem de ser possível estimar o esforço necessário para a sua realização; **(3)** cada *User Story* tem de ter a capacidade de ser testada com o intuito de verificar se está de acordo com as expectativas comportamentais dos *stakeholders* e por último, **(4)** cada *User Story* é alvo de priorização por parte dos *stakeholders* para que a equipa de desenvolvimento se consiga focar nas *User Stories* prioritárias atempadamente [MA04]. A Figura 2.3 mostra um possível *template* de uma *User Story*.

Titulo: Cliente levanta dinheiro
Como um Titular de Conta,
Eu quero levantar dinheiro de um ATM,
Para que não tenha de esperar na fila do banco.

Figura 2.3: *Template (Role-feature-reason)* de *User Story*

De notar que uma *User Story* é ponto de partida para a construção de um requisito. A partir de uma *User Story* nascem tarefas que expressam com mais detalhe as expectativas dos *stakeholders* e serão verificadas com recurso a testes de aceitação. As tarefas são descobertas através de discussões com os *stakeholders*, onde é debatido com mais certeza as necessidades dos mesmos. A Figura 2.4 mostra um subconjunto dos artefactos de requisitos presentes num processo ágil [Lef11]. Esta dissertação foca-se na construção de testes de aceitação a partir de cenários comportamentais *Behavior-Driven Development* (secção 3.4), que representam os comportamentos de uma *User Story*.

A necessidade da presença dos *stakeholders* no início de cada ciclo de desenvolvimento para a construção e discussão de *User Stories* pode resultar num problema caso haja a

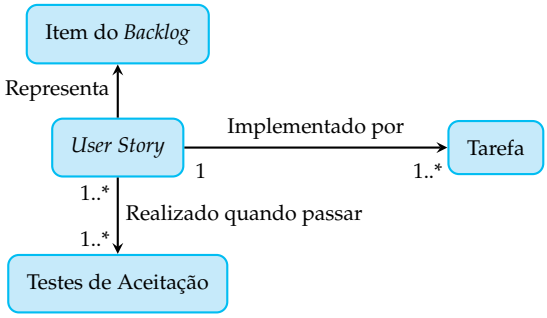


Figura 2.4: Subconjunto de artefactos de requisitos ágeis e as suas relações [Lef11]

indisponibilidade dos mesmos [CR08]. Muitas das vezes, apenas um representante dos *stakeholders* está disponível. Quando existem múltiplos *stakeholders*, o seu consenso (ou falta dele) e confiança na equipa de desenvolvimento tem um relacionamento direto com a eficácia das discussões.

Impacto na equipa de desenvolvimento

As *User Stories*, apesar de não serem criadas com um fim documentativo, são debatidas regularmente no seio da equipa de desenvolvimento. Isto resulta numa validação constante de requisitos, devido à comunicação continua, e num melhor relacionamento dos membros da equipa [GS09]. Outro estudo de [Abd+11] evidenciou que os membros da equipa: **(1)** sabiam as *User Stories* de cada membro; **(2)** contribuíam para resolver problemas de colegas e **(3)** cada membro tinha uma conceção do sistema que ia evoluindo ao longo das reuniões e do desenvolvimento. Portanto, a fonte principal de toda a informação relativa ao sistema em termos de funcionalidades e evolução é adquirida através de colaboração de *stakeholders* e membros da equipa de desenvolvimento [SRP09].

Assim, a compreensão de ambos os interlocutores é essencial para que a equipa de desenvolvimento perceba as necessidades dos *stakeholders* e consiga entregar *software* que as satisfaça corretamente. Para demonstrar um caminho possível a fim de melhorar a comunicação entre *stakeholders* e a equipa de desenvolvimento, esta dissertação tem o objetivo de conseguir melhorar a interação de **todos** os *stakeholders* através de modelos cognitivos, especificamente mapas mentais.

2.3.2 Modelos Cognitivos

A introdução desta secção nasce com o propósito de apresentar ao leitor um modelo cognitivo, em concreto, um mapa mental. Este será o modelo utilizado para tentar aproximar os *stakeholders* no processo de captura de requisitos no contexto do AM.

Um mapa mental é um modelo cognitivo com o objetivo de representar conhecimento [BB93]. É caracterizado por ter uma imagem no seu centro, para captar a atenção do leitor com o tema principal do diagrama. Desse centro, nascem ramos com uma frase (ou imagem) chave que representa um subtema relacionado. Desses subtemas podem nascer

outros e assim sucessivamente. Num todo, fica com uma forma radial e no centro fica o tema principal do diagrama. A Figura 2.5 exemplifica um mapa mental.

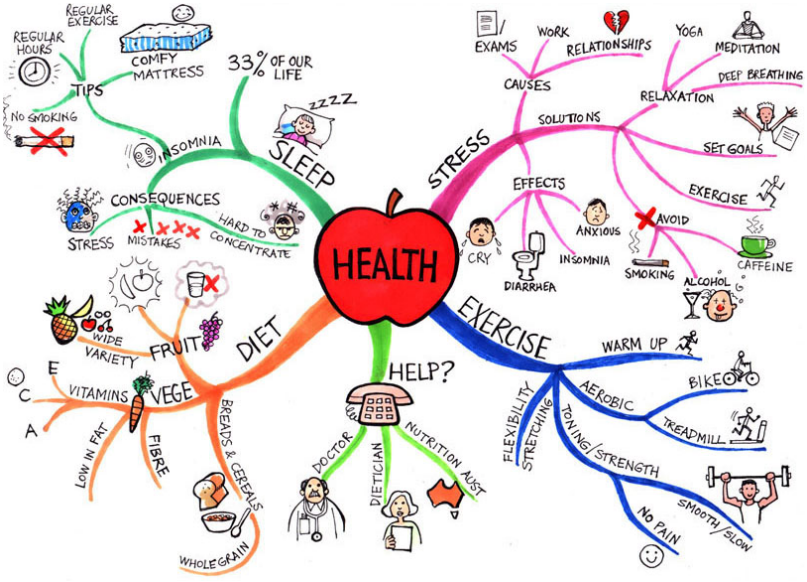


Figura 2.5: Mapa mental [Buz14]

O criador dos mapas mentais, *Tony Buzan*, iniciou esta pesquisa para responder a um conjunto de perguntas: *Como é que nós aprendemos a aprender? Quais são as melhores técnicas para memorizar? Pensar? Ler?*. Iniciou a sua pesquisa estudando o cérebro, estudando como reage a diversos estímulos. O cérebro é constituído por dois hemisférios, o esquerdo e o direito. A parte esquerda é onde é exercitada a lógica (números, palavras, sequências). A direita é onde são estimulados elementos visuais, como a cor, imagens, associações, padrões e formas.

Buzan concluiu que, utilizando ambos os hemisférios do nosso cérebro, conseguimos captar e reter mais informação. Dado isso, criou o mapa mental, que estimula ambos os hemisférios do nosso cérebro de forma a aumentar a nossa capacidade de aprender e pensar sobre o assunto.

Existem trabalhos relacionados com o uso de mapas mentais no espetro de engenharia de requisitos. *Mahmud* e *Laplante* relatam trabalhos em que se utilizaram mapas mentais para representar requisitos, auxiliando na captura dos mesmos, de moda a facilitar discussões com *stakeholders* [MV11; Lap07]. De acordo com *Alexander*, os mapas mentais podem ajudar engenheiros de requisitos a lembrarem-se e terem uma visualização geral dos detalhes do sistema devido ao uso de palavras e notações visuais especificas [MA04]. Foi também proposto o uso de mapas mentais para decisões feitas colaborativamente, criando uma estrutura de dados baseada no mesmo [ZAS14].

Esta dissertação tem como objetivo utilizar mapas mentais de forma a facilitar a partilha e a representação de informação entre os *stakeholders* e a equipa de desenvolvimento, a fim de capturar requisitos de *software*.

2.4 Resumo

Neste segundo capítulo foram introduzidos dois conceitos alicerces desta dissertação: a Modelação Ágil e a Engenharia de Requisitos. A Modelação Ágil consiste nos princípios e valores que as metodologias ágeis e as suas práticas se baseiam. Já a disciplina de Engenharia de Requisitos é caracterizada pela forma de manter, organizar, validar e capturar requisitos no desenvolvimento de *software*. Nas práticas ágeis, as *User Stories* são omnipresentes durante o processo de captura e validação de requisitos e tem um impacto positivo na equipa de desenvolvimento promovendo a discussão do sistema entre todos os *stakeholders* envolvidos. As metodologias ágeis abordadas, SCRUM e o XP, pressupõem um envolvimento dos *stakeholders* e um desenvolvimento iterativo da aplicação, com validações no final de cada iteração a fim de ter um *feedback* permanente. No próximo capítulo iremos explorar o desenvolvimento baseado em cenários, a sua relação com testes de aceitação e o *Behavior-Driven Development*.

3

Desenvolvimento baseado em cenários

Este capítulo tem como objetivo guiar o leitor no desenvolvimento baseado em cenários. Este tema está intimamente relacionado com a construção de testes de aceitação a partir de requisitos capturados através de *User Stories*. Nos processos ágeis, um requisito diz-se *satisfeito* após passar nos seus testes de aceitação. Esta dissertação tem como objetivo melhorar aspectos de validação e verificação na construção de testes BDD com a utilização de mapas mentais. Para conseguir compreender o BDD, é necessário a compreensão dos seus antecessores, Desenvolvimento Orientado a Testes e Desenvolvimento Orientado a Testes de Aceitação.

Na primeira parte deste capítulo vamos apresentar uma visão geral de testes no contexto ágil. De seguida, apresentaremos o Desenvolvimento Orientado a Testes, em que consiste e os seus benefícios e desafios. Em terceiro lugar, abordaremos a evolução do Desenvolvimento Orientado a Testes para o Desenvolvimento Orientado a Testes de Aceitação. Finalmente, vamos expor o *Behavior-Driven Development*, onde explicaremos todo o seu processo e os ganhos obtidos com a sua evolução do Desenvolvimento Orientado a Testes de Aceitação.

3.1 Testes em processos ágeis

Nos processos ágeis, os testes podem ser agrupados em quatro grupos distintos, como demonstra a Figura 3.1 [Lef11]. No primeiro quadrante (Q1), fazem parte os testes de *caixa-branca*, que são testes unitários a componentes internos do sistema. São testes automatizados porque normalmente são em grande quantidade. No segundo quadrante

(Q2), pertencem os testes considerados de *caixa-negra*, i.e., dizem respeito a funcionalidades do sistema. Servem para verificar se o sistema faz o que é suposto em termos de requisitos dos *stakeholders*. Já sobre o terceiro quadrante (Q3) recaem testes manuais do sistema como um todo. São testados aspectos de usabilidade com utilizadores finais e várias variações de possíveis cenários. Finalmente, no quarto quadrante (Q4) estão testes relativos a aspectos de requisitos de qualidade, como testes de *stress*, de segurança e de *performance*. Estes são feitos através de ferramentas especializadas.

Testes **Funcionais** **Q2**	**Q3** Testes de **aceitação do sistema**
Testes **Unitários** **Q1**	**Q4** Testes de **Qualidade**

Figura 3.1: Diferentes tipos de testes em processos ágeis [Lef11]

Esta dissertação irá debruçar-se em *testes funcionais* integrados no segundo quadrante. Os testes de aceitação são uma forma de validação de *User Stories* e consequentemente uma forma de garantir que o sistema corresponde às expectativas dos *stakeholders* [Kos07]. Os testes criados, são mantidos inalterados, para caso exista alguma alteração no sistema se consiga verificar se ocorreu algum efeito nefasto no mesmo. A criação dos testes é feita através de discussões tendo como base uma *User Story* e são escritos utilizando a linguagem dos *stakeholders*.

O *Behavior-Driven Development* é uma prática ágil de captura e criação de testes de aceitação, mas tem como alicerces o *Test-Driven Development* e o *Acceptance-test Driven Development* [SW11]. As próximas secções vão explicar a evolução desde o *Test-Driven Development* até ao *Behavior-Driven Development*, explorando os aspectos que podem ser melhorados no BDD.

3.2 Desenvolvimento Orientado aos Testes

O Desenvolvimento Orientado aos Testes, em inglês (*Test-Driven Development*- TDD), é uma prática de desenvolvimento de *software* onde é escrito um teste unitário com resultado insatisfatório antes da realização de código de produção. Esta prática foi principalmente adotada nos processos de desenvolvimento ágil, nascendo do XP. Sugere que os testes sejam escritos no início do desenvolvimento e não no final como na forma tradicional de desenvolvimento de software. Assim, os programadores passam a ter controlo sobre o que está a funcionar corretamente no sistema e consequentemente a qualidade do código produzido aumenta [Bec02].

Apesar de a prática ter um pouco mais que uma década, o seu processo de implementação continua confuso e suscita muitas dificuldades a iniciantes [HU12]. O guia de utilização do TDD apresentado por [Bec02] consiste em:

1. Escrever um novo teste;
2. Correr todos os testes e verificar se o último falha;
3. Escrever código para fazer o novo teste passar;
4. Correr de novo todos os testes e verificar se todos passam;
5. Fazer refactorização de código para remover duplicação.

A Figura 3.2 ilustra o ciclo da prática com o uso das cores (Vermelho, Verde) associada a cada fase: o vermelho significa o insucesso de um teste; o verde, por outro lado, significa que o teste passou, de preferência com o mínimo de código possível; por fim a refactorização é utilizada para eliminar qualquer duplicação de código ou más práticas de programação utilizadas para fazer o teste passar. O ciclo repete-se até todos os requisitos estarem satisfeitos.

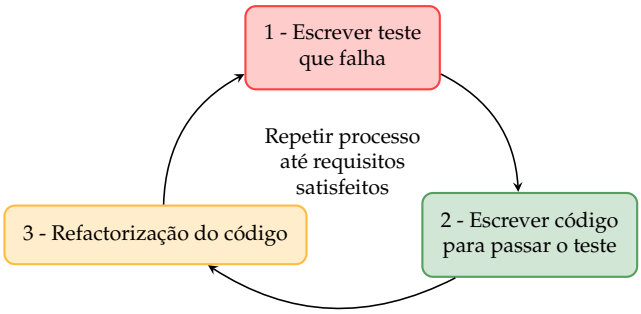


Figura 3.2: Ciclo TDD [EMJ10]

3.2.1 Benefícios e desafios

Durante a última década, foram feitos vários estudos que demonstram os benefícios de TDD, onde se concluíram provas de aumento de qualidade interna [Shu+10; EMJ10] e de acordo com um *survey*[1], 196 empresas classificam o TDD como a prática com mais influência para o sucesso. Os praticantes de TDD afirmam que, com a construção sistemática de testes, ficam a conhecer o sistema com mais facilidade e, simultaneamente, criam documentação sobre o mesmo.

Apesar de existirem alguns indícios que o TDD possa vir a melhorar a qualidade do *software*, o percurso até ao melhoramento é moroso. De acordo com [HU12], iniciantes de TDD têm de ultrapassar vários obstáculos até dominarem a prática. A resistência à mudança de hábitos de programação, a falta de *design* prévio e a incapacidade de tomar decisões sobre o mesmo são algumas das dificuldades presentes. Também existem casos de pura dificuldade em iniciar o processo. Como consequência dessas advertências iniciais, surgem efeitos na produtividade como demonstram estudos [EMJ10] e provocam insegurança nalgumas empresas na iniciação da prática.

[1] *Cutter Consortium*

Não obstante aos seus benefícios e desafios, o TDD por si só está muito focado em testes [Fra+03; Mad10]. Existem questões a responder como: *estamos a desenvolver a funcionalidade correta?*; *O conjunto de testes criados validam a minha funcionalidade?*. O TDD, *per se*, não proporciona nenhuma abordagem para verificar se o que está a ser testado corresponde a uma necessidade dos *stakeholders* e, mesmo que corresponda, não existe forma de verificar quando finalizar a construção de testes.

3.3 Desenvolvimento Orientado a Testes de Aceitação

O Desenvolvimento Orientado a Testes de Aceitação, em inglês (*Acceptance-test Driven Development*- ATDD), é um processo de captura e verificação de requisitos [Kos07]. Foi evoluído do TDD, com o objetivo de criar testes que validem regras de negócio dos *stakeholders*. Com a criação dos testes de aceitação, existe uma validação das necessidades dos *stakeholders* e o progresso do desenvolvimento é feito com base nos testes de aceitação satisfeitos.

3.3.1 Ciclo de desenvolvimento

O processo de criação de testes de aceitação (representado na Figura 3.3) é composto por quatro passos: escolha de uma *User Story* a realizar o testes, o desenho dos testes, a automatização dos mesmos e, por fim, a implementação que valide os testes [Kos07]. O ciclo repete-se ao longo da iteração desde que existam *User Stories* a implementar. Em

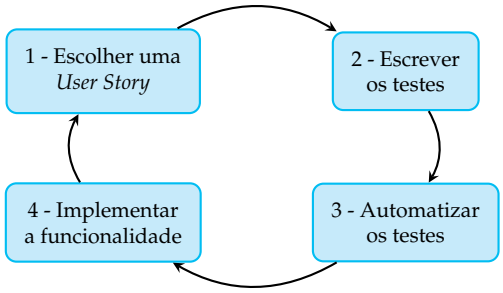


Figura 3.3: Processo de ATDD [Kos07]

mais detalhe, cada passo do processo consiste em:

1. **Escolher uma *User Story***: é escolhida uma *User Story* por implementar para serem criados testes de aceitação. A escolha da mesma deve ser feita com base na sua prioridade para os *stakeholders* envolvidos. Como nas metodologias ágeis o desenvolvimento é iterativo, o desenvolvimento começa com as necessidades mais cruciais para o negócio dos *stakeholders*.
2. **Escrever os testes**: com a *User Story* selecionada, é feita uma reunião com os *stakeholders* a fim de criar os testes de aceitação. Estes, ao serem escritos, têm um conjunto de propriedades a serem respeitadas: **(1)** são escritos pelos *stakeholders*,

visto que eles é que sabem os critérios de aceitação, mas estão também presentes membros da equipa de desenvolvimento, juntamente com *testers*, para ajudar a dominar a prática e a escrita dos testes; **(2)** são escritos delineando um conjunto de passos abstratos, escondendo como será produzido e focando-se apenas no comportamento; **(3)** cada teste escrito é feito utilizando a linguagem de domínio da aplicação, sendo esta uma propriedade fundamental, para que os *stakeholders* participem na construção dos testes; **(4)** cada teste deve ser conciso, preciso e inequívoco de forma a que seja fácil o seu mapeamento para código.

3. **Automatizar os testes**: o terceiro passo é a automatização dos testes escritos. Isto serve para verificar a qualquer altura se o sistema passa ou falha nas especificações dos testes. Normalmente, são utilizadas ferramentas FIT[2].

4. **Implementar a funcionalidade**: após a criação dos testes, a equipa de desenvolvimento começa a produção de código com o intuito de satisfazer o teste. Quando o código produzido verifica os testes de aceitação, a *User Story* está implementada com sucesso no sistema.

3.4 Desenvolvimento Orientado ao Comportamento

O Desenvolvimento Orientado ao Comportamento, em inglês (*Behavior-Driven Development*- BDD), é uma prática de desenvolvimento ágil que evoluiu do *Test-Driven Development* e do *Acceptance-test Driven Development* [SW11]. No BDD, a palavra principal passa a ser *comportamento* em vez de *teste*. A razão pela qual o BDD nasceu foi para clarificar a forma mais correta de utilizar TDD e o ATDD por parte da equipa de desenvolvimento: saber como os testes surgem, o porquê da sua existência e qual o seu papel no *software* a construir [Nor13]. A principal diferença é que com o BDD, os comportamentos (que dão origem a testes) são especificados diretamente com os *stakeholders* através de cenários BDD. Assim, o BDD parte do ATDD com o objetivo de obter a descrição do comportamento do sistema, tirando proveito de uma comunicação ativa com os *stakeholders*. Com a informação obtida, são criados testes de aceitação para a verificação do comportamento esperado.

A Figura 3.4, baseada em [SW11; Nor13], exemplifica a evolução do BDD. Tal como a figura indica, o BDD nasceu para incorporar os *stakeholders* no processo de desenvolvimento, aperfeiçoando a prática orientada aos testes. Cria, deste modo, uma abstração para que os *stakeholders* façam parte do processo de desenvolvimento. Assim, o BDD é caracterizado por ser uma prática de ER, e, ao mesmo tempo, de construção de Testes de Aceitação. De forma a melhorar a integração dos *stakeholders* no processo de desenvolvimento, o BDD implementa práticas de Desenho Orientado ao Domínio, tema que vai ser explicado na secção seguinte.

[2]FIT- *Framework Integrated Test*

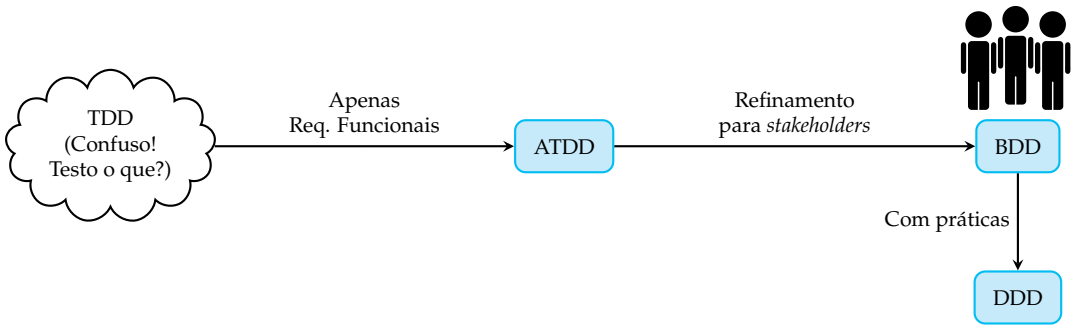


Figura 3.4: Evolução de práticas Test-Driven Development [SW11; Nor13]

3.4.1 Desenho Orientado ao Domínio

Qualquer aplicação está diretamente ligada ao domínio na qual vai ser aplicada. A informação sobre esse mesmo domínio está presente e espalhada entre diversos *stakeholders*. A necessidade de aproximação dos *stakeholders* com a equipa de desenvolvimento vem da importância da informação que os *stakeholders* têm sobre esse domínio. Com a compreensão dessa informação, são capturados os requisitos dos *stakeholders* e ao implementá-los satisfazemos as necessidades dos *stakeholders*. O Desenho Orientado ao Domínio, em inglês (*Domain-Driven Design*- DDD), é uma filosofia de *design* [Eva03] que tenta resolver o problema do *gap semântico* entre o domínio das equipas de desenvolvimento e dos *stakeholders* com a utilização de um modelo de domínio ao longo do desenvolvimento.

No DDD, existem dois componentes essenciais: um modelo de domínio e uma linguagem ubíqua. O modelo de domínio é uma simplificação da realidade. É uma interpretação da realidade que abstrai os aspectos irrelevantes, restando apenas os significativos [Eva03]. A informação capturada no modelo está organizada de forma a permitir tomar decisões que ajudem a resolver o problema. Essa informação, é utilizada nas fases de implementação onde os nomes dos objetos, das suas variáveis e relações, são retirados do modelo de domínio previamente desenhado. Isso permite que a equipa de desenvolvimento consiga ter uma manutenção facilitada e permite que exista um ponto comum de linguagem utilizado por parte da equipa. Um exemplo modelo de domínio é apresentado na Figura 3.5.

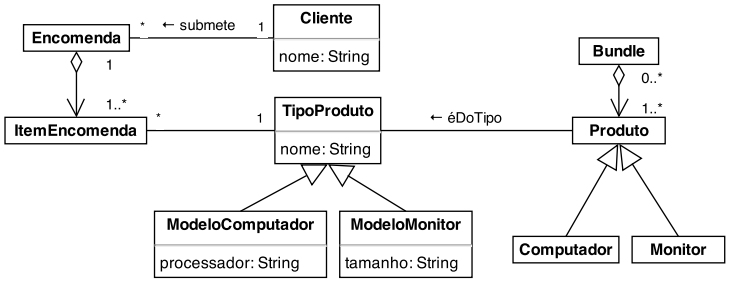


Figura 3.5: Exemplo de um modelo de domínio traduzido [Eng+05]

O exemplo apresentado na Figura 3.5 retrata um problema típico de encomenda de produtos. A notação típica utilizada na representação de modelos assemelha-se à notação do UML. Este modelo representa as entidades necessárias para que um *Cliente*, consiga criar uma *Encomenda*, para encomendar produtos com dois tipos, *ModeloComputador* e *ModeloMonitor*. Este modelo apresenta algum refinamento porque apresenta cardinalidade, herança e associação, no entanto tipicamente os modelos começam apenas com as entidades do problema e a suas relações.

O outro componente do DDD é a linguagem utilizada no processo de desenvolvimento. Os *stakeholders* envolvidos no desenvolvimento do *software* possuem a sua própria linguagem e podem ter dificuldade em perceber termos técnicos próprios do desenvolvimento de um *software*. Caso isso aconteça, tem de existir uma tradução de termos técnicos entre a equipa de desenvolvimento e os *stakeholders*, que pode resultar em perdas de informação e, consequentemente, numa linguagem fraturada. Isso pode resultar em implementações erróneas que pioram a qualidade do *software*. Para combater essa situação, o DDD propõe a construção de uma linguagem ubíqua, uma linguagem única do projeto. Uma linguagem ubíqua consiste numa linguagem intimamente ligada com o modelo de domínio, i.e., que expresse os conceitos mapeados no modelo de domínio. Caso exista algum conceito mencionado em discussões no desenvolvimento do projeto e esse não esteja mapeado no modelo de domínio, existe uma discordância entre ambos que deve ser reparada no modelo de domínio.

De forma a construir um modelo de domínio, existem quatro componentes fundamentais [Eva03]: **(1)** uma entidade que representa um objeto, tendo este uma identidade única; **(2)** Objetos de Valor que não tem identidade única (características de algo); **(3)** Serviços que são operações que não estão ligadas diretamente a algum objeto e **(4)** módulos que representam um conjunto de objetos de conceitos relacionados. A Figura 3.6 mostra o relacionamento entre os conceitos de domínio e a construção de uma linguagem ubíqua e de um modelo de domínio.

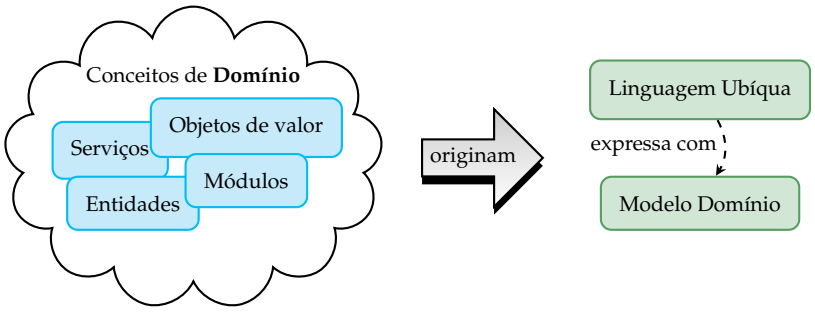


Figura 3.6: Relacionamento entre conceitos de domínio e o modelo de domínio e a linguagem ubíqua. Adotado de [Eva03]

O BDD adota uma linguagem ubíqua em todo o processo (3.4.2). Essa linguagem é única ao processo, contém conceitos e expressões próprias dos *stakeholders*, para tentar minorar o *gap semântico* entre as discussões. O BDD constrói essa linguagem através da

comunicação contínua com os *stakeholders* ao longo do seu processo e utiliza-a em todos os seus artefactos.

3.4.2 Processo do Desenvolvimento Orientado ao Comportamento

O processo do BDD é composto por seis passos, como mostra a Figura 3.7. É um processo para ser inserido numa metodologia ágil, logo é iterativo. Após o passo número 6, são validados os artefactos produzidos com os *stakeholders* e o ciclo é reiniciado onde os artefactos anteriormente produzidos podem sofrer alterações. O aspecto mais importante do *Behavior-Driven Development* é a comunicação com os *stakeholders* e os frutos que trazem para o desenvolvimento.

É com a comunicação que se mantêm desde o planeamento até a fase de análise que nasce a linguagem do projeto, constantemente atualizada e refletida na implementação. Os cenários BDD, que representam o comportamento esperado do *software*, são escritos utilizando essa mesma linguagem. Os testes de aceitação são criados a partir dos cenários comportamentais e, por sua vez, partilham a mesma linguagem, trazendo, assim, a linguagem dos *stakeholders* para o código da aplicação. Através deste processo, é criada uma linguagem do projeto com entidades, relações e conceitos do domínio dos *stakeholders*.

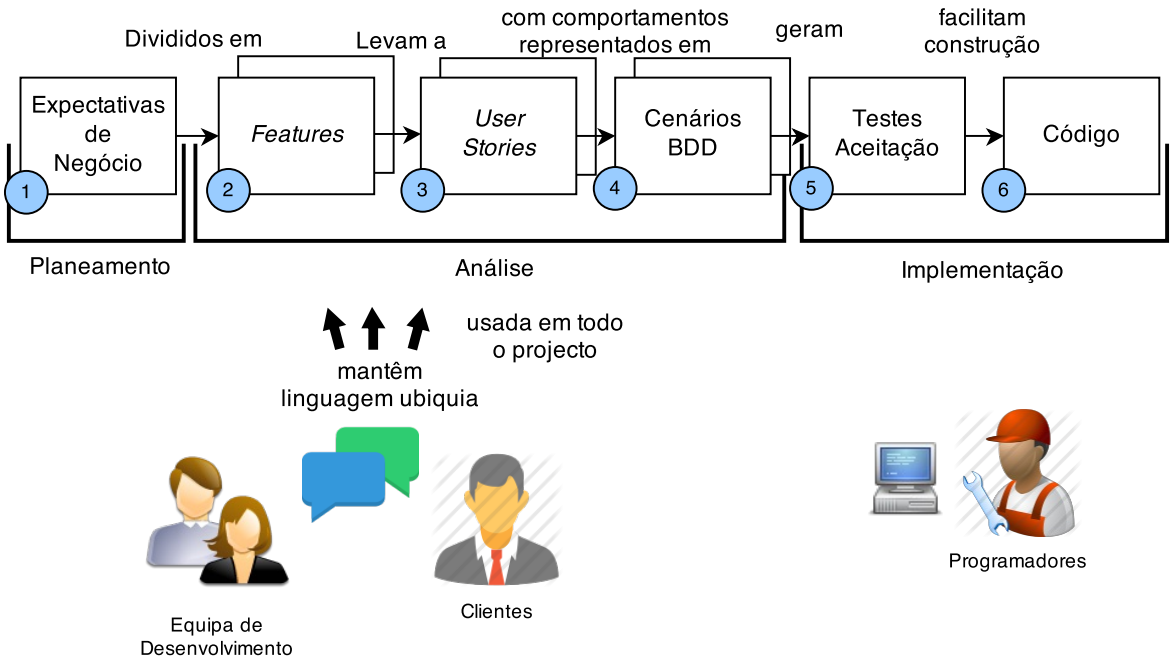


Figura 3.7: Processo do *Behavior-Driven Development*

Tal como a Figura 3.7 indica, a prática BDD inicia-se com a comunicação, entre a equipa de desenvolvimento e os *stakeholders*. Dessa comunicação, na fase de planeamento, são definidas expectativas de negócio. Os comportamentos são derivados dos objetivos de negócio que se pretendem produzir. Na fase de análise, os objetivos de negócio são decompostos em funcionalidades (*features*) e cada uma das funcionalidades está

associada a uma ou mais *User Stories*.

Cada *User Story* é instanciado com múltiplos cenários BDD, onde em cada um é delineado o comportamento esperado para aquela situação. O cenário é construído com recurso à linguagem do projeto e é escrito pelos *stakeholders* ou em conjunto com a equipa de desenvolvimento. Os cenários BDD têm um *template*, também escrito por uma linguagem ubíqua independente de domínio, de forma a ser utilizado em qualquer situação. A Figura 3.8 mostra o *template* de cenário BDD.

Título: (Título para descrever *story*)
Como um (papel)
Eu quero (funcionalidade)
Para que (benefício)

Cenário 1 (título)
Dado (contexto)
E (outro contexto)
Quando (evento)
Então (resultado)

Figura 3.8: *Template* de cenário BDD [Nor13].

Um cenário BDD é composto por uma parte inicial que representa uma *User Story* (secção 2.3.1). A segunda parte do cenário descreve o comportamento em si. Dado um contexto inicial, descrito por várias características, quando, eventualmente, ocorrer uma mudança no domínio, então obtém-se o estado final pretendido. O objetivo de um cenário BDD é descrever como o sistema implementa uma funcionalidade e como se deve comportar dado um contexto e a ocorrência de um certo evento. A Figura 3.9 mostra um exemplo de cenário BDD [Nor13]:

Titulo: Cliente levanta dinheiro
Como um Titular de Conta,
Eu quero levantar dinheiro de um ATM,
Para que não tenha de esperar na fila do banco.

Cenário 1: Cartão desactivado
Dado um cartão desactivado
Quando o cliente requisitar $20
Então o ATM deve reter o cartão

Figura 3.9: Exemplo cenário BDD.

Após a construção de cenários comportamentais, os mesmos serão transformados em testes de aceitação. A transformação é feita através de regras de mapeamento, para que o código seja escrito com a linguagem do cenário. A Listagem 3.1 exemplifica a construção de classes de teste e uma possível solução para o exemplo da Listagem 3.9. O nome dos métodos e classes devem respeitar os nomes escritos no cenário BDD. Assim, a manutenção do sistema fica facilitada visto que tem nomes explícitos e associados ao seu comportamento esperado. Com a construção automática de testes de aceitação, a fase de implementação tem um início mais robusto, visto que os programadores já têm como

base cenários de comportamento validados por parte dos clientes.

Listagem 3.1: Exemplo de classe de teste na plataforma *JBehave*

```
public class CashWithdrawingSteps {
  private ATM atm;
  private Cartao cartao;
  private BigDecimal retorno;
  private Throwable throwable;

  @Given("um cartao desactivado")
  public void CartaoDesactivado() {
    cartao = new Cartao();
    cartao.setInvalid();
    atm = new ATM();
  }
  @When("o cliente requisitar $quantia")
  public void clienteRequisitarDinheiro(@Named("quantia") BigDecimal quantia) {
     try {
      retorno = atm.levantar(cartao, quantia);
    } catch (CardRetainedException exception) {
      throwable = exception;
    }
  }
  @Then("o ATM deve reter o cartao")
  public void atmDeveReterCartao() {
    Assert.assertNull(retorno);
    Assert.assertTrue(throwable instanceof CardRetainedException);
  }
}
```

Desafios

Existem poucos estudos sobre uma avaliação da utilização de BDD. No entanto, alguns autores [Lop11; SW11] reuniram benefícios e desafios com a mesma. No trabalho de Lopes[Lop11], foi desenvolvida uma aplicação utilizando BDD e o mesmo constatou que é moroso iniciar o desenvolvimento, devido à dificuldade da utilização das ferramentas BDD. Lopes constatou também que o processo do BDD não explica como abordar *bugs* da aplicação. O procedimento utilizado por Lopes foi: cada *bug* encontrado era tratado como uma *User Story* de alta prioridade, a fim de ser reparado rapidamente. Outro problema encontrado dizia respeito às decisões de *design* serem individuais e não existir forma de partilhar informação do domínio. Isto podia dificultar o processo de desenvolvimento e produzir uma arquitectura incapaz e não documentada. Outro grande contratempo sentido, partilhado com o uso de *User Stories*, é a necessidade de um *stakeholder* para conseguir produzir cenários e iniciar o desenvolvimento.

Como já foi dito anteriormente, o BDD é uma técnica de validação e verificação de

requisitos, mas não contempla requisitos de *User Interface* nem consegue expressar requisitos *crosscuting*. As ferramentas atuais focam-se apenas na implementação e não existe nenhum apoio de *design* e planeamento com toda a informação obtida pelos cenários.

Uma das contribuições desta dissertação é melhoramento do apoio na construção dos cenários BDD. O apoio é através duma DSL para representar cenários BDD, composta por regras de validação sintática para impedir a construção de cenários incorrectos e a capacidade de incorporar informação proveniente de um modelo de domínio. Com a introdução dum modelo de domínio, também com suporte duma DSL, no processo do BDD, abordamos o problema de escassez dum artefacto para agrupar e partilhar informação relevante de domínio. Por último, pretendemos melhorar a criação dos casos de testes, com uma transformação automática a partir da DSL.

3.5 Resumo

Este capítulo da dissertação abordou três temas relacionados com o Desenvolvimento Baseado em Testes: o TDD, o ATDD e o BDD. Sendo o BDD considerado uma evolução do ATDD que, por sua vez, é considerado uma evolução do TDD estes estão todos relacionados, onde a principal diferença é a continua abstração para integrar os *stakeholders* no desenvolvimento dos testes com o intuito de capturar e validar requisitos funcionais. No entanto, o BDD, apesar de ter melhorado a abstração com o uso de práticas de DDD, ainda tem alguns desafios como: a partilha de informação capturada nos cenários BDD com o resto da equipa de desenvolvimento; um moroso início de desenvolvimento dadas as tecnologias actuais; a inexistência de uma *big picture* dos requisitos funcionais para auxiliar a construção de uma arquitectura. No próximo capítulo, serão introduzidos conceitos de MDD e DSL. Essas são as técnicas que utilizámos para a construção da nossa solução.

4

Desenvolvimento Orientado a Modelos e Linguagens Específicas de Domínio

O Desenvolvimento Orientado a Modelos, em inglês (*Model-Driven Development*- MDD), é uma metodologia de desenvolvimento de *software* em que os modelos são promovidos para artefactos principais no desenvolvimento, sofrendo constantes transformações até à geração de código. Os mesmos guiam o desenvolvimento, onde cada modelo criado representa uma visão característica sobre um aspecto relativo à construção de *software*.

Neste capítulo são abordados aspectos relativos ao MDD e a sua importância na construção de uma DSL. O processo de desenvolvimento de uma DSL é abordado, detalhando as suas vantagens e ferramentas de apoio à sua construção.

4.1 O que é um modelo?

Um modelo é uma representação abstrata de um sistema que permite fazer inferências e previsões sobre o mesmo [Küh06]. Existem três características predominantes que o tornam útil no desenvolvimento: mapeamento de um sistema, redução de características insignificantes e apenas deve ser criado com propósito de ajudar a guiar a construção da solução do sistema.

Ao longo das diversas fases de desenvolvimento, existem múltiplos modelos com um propósito de representar diversos aspectos do sistema. Para que a informação capturada em diversos modelos, numa determinada fase, permaneça na fase seguinte, são utilizadas

transformações de modelos. As transformações garantem a coesão e a coerência entre modelos. Apesar de ainda existirem muitos problemas nesta área, como a rastreabilidade de componentes e a atualização de todos os mesmos face a modificações, é uma estratégia de desenvolvimento muito utilizada para projetos de grande dimensão com altos riscos envolvidos.

Na secção 4.2 será abordado o tema de Meta-Modelos. Os Meta-Modelos definem a construção de modelos e como esses conseguem garantir a semântica desejada e abranger apenas um domínio específico pensado para um certo grupo de utilizadores.

4.2 Meta-Modelos

Os Meta-Modelos são modelos para definir modelos [OMG14]. Servem para definirmos objetos, os seus elementos e as ligações entre eles. A partir de um Meta-Modelo, conseguimos instanciar modelos que respeitam as suas regras e restrições. A Figura 4.1 demonstra como são criadas linguagens a partir de meta-modelação e a criação de modelos para modelar sistemas. Já os Meta Meta-Modelos definem-se a eles próprios e são utilizados para especificar Meta-Modelos.

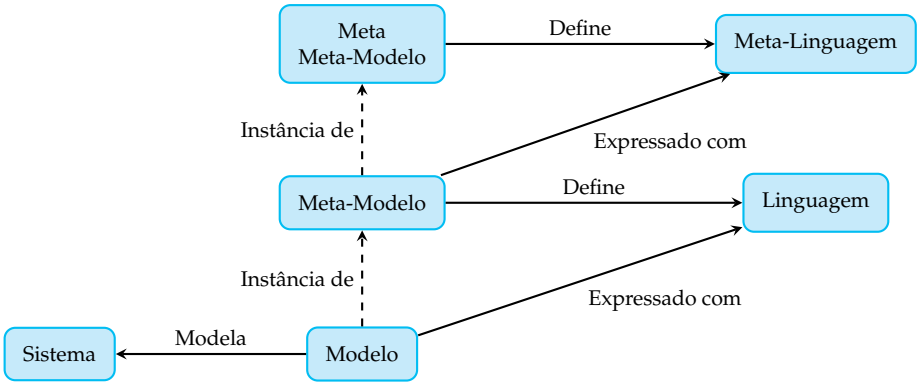


Figura 4.1: Esquema de definição de Linguagens [Küh06]

4.3 Transformações de modelos para modelos

Visto que num desenvolvimento MDD, os modelos são considerados artefactos que guiam o desenvolvimento através de sucessivas transformações, é necessário compreender como são realizadas. As transformações são especificadas tendo em conta os meta-modelos dos modelos de entrada e saída. São feitas correspondências entre os elementos de entrada e saída. Tendo em conta a Figura 4.2, uma transformação do um *Modelo a* para um *Modelo b* necessita de três componentes: um meta-modelo para os modelos **a**, **b** e para a transformação. A transformação em si é definida num modelo onde são feitas as correspondências entre os elementos dos modelos **a** e **b** e garantida uma semântica entre os mesmos.

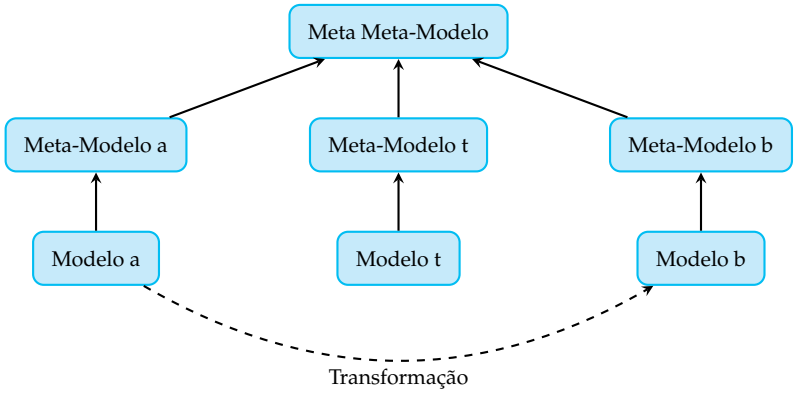


Figura 4.2: Esquema de transformação entre Modelos

4.3.1 Atlas Transformation Language

Uma possível solução para definir as regras de transformação de modelos para modelos é a linguagem *Atlas Transformation Language* (ATL) [Ecl14]. O ATL é uma linguagem de transformação Modelo-Para-Modelo. A transformação tem como componentes três partes: o modelo de entrada, a transformação e o modelo de saída. A linguagem recorre aos Meta-Modelos do modelo de entrada e saída e faz simplesmente uma correspondência entre cada objeto e relação. Um objeto pertencente ao modelo de entrada vai corresponder a outro objeto do modelo de saída. As regras de transformação podem ser escritas de forma imperativa ou declarativa e são impostas pelo criador do *script* de transformação. O ATL permite também a pesquisa, navegação e modificação dos modelos envolvidos na transformação.

4.3.2 Epsilon Transformation Language

Outra linguagem de transformação, com as características necessárias no contexto desta dissertação é o *Epsilon Transformation Language* (ETL). Tal como o ATL, é também uma linguagem híbrida, baseada em regras de transformação de modelo para modelo. Pertence a uma família de linguagens baseadas no *Epsilon Object Language* (EOL). O ETL funciona também com base em regras de mapeamento permitindo a capacidade e modificação dos modelos envolvidos na transformação e transformar vários modelos de entrada num de saída. Esta será a linguagem escolhida para executar as transformações na dissertação porque o autor já esta familiarizado com a mesma e recorrerá a uma outra linguagem, o *Epsilon Generation Languagem* (EGL), baseada também no EOL. A linguagem ETL servirá para realizar as transformações dos vários cenários BDD num modelo de domínio.

O EGL é uma linguagem de transformação de Modelos-Para-Texto, que transforma um modelo de entrada para texto. A finalidade desta transformação será a construção dos testes de aceitação para a plataforma *JUnit*. As regras de mapeamento funcionam de forma semelhante à de uma transformação Modelo-Para-Modelo. A diferença recai em que não existe um modelo de saída, mas sim texto e assim não é validado por nenhum

Meta-Modelo. As listagens 4.1 e 4.2 mostram um pequeno exemplo de um transformação utilizando o ETL e EGL respectivamente.

Listagem 4.1: Exemplo de uma transformação ETL [Fou12b]

```
rule Tree2Node
  transform t : Tree!Tree
  to n : Graph!Node {

  n.name = t.label;
  if (t.parent.isDefined()){
    var e : new Graph!Edge;
    e.source ::= t.parent;
    e.target = n;
  }
}
```

Listagem 4.2: Exemplo de uma transformação EGL [Fou12a]

```
[% for (entry in dataType.getEAnnotation("doc").
    getDetails().select(e|e.key="description")) {%]
    <hr />
    <p id="description">
      [%=entry.value%]
    </p>
[% } %]
```

4.4 Linguagens Especificas de Domínio

As Linguagens Especificas de Domínio, em inglês (*Domain Specific Language*-DSL), são, tal como o nome indica, linguagens não genéricas que descrevem mais proximamente um determinado domínio. Um exemplo de uma DSL é o *CSS* [1]. Ao contrário de linguagens para propósito geral, estas são construídas a pensar num subconjunto de utilizadores, a pensar nas suas necessidades e domínios de trabalho. Assim, uma DSL consegue oferecer melhor usabilidade, minimizando erros dos utilizadores e consequentemente aumentar produção [Voe+13]. Nesta dissertação foram criadas duas DSLs, uma dedicada à construção de modelos de domínio, que serve para agrupar informação relevante sobre o domínio da aplicação a desenvolver e outra para cenários BDD, visando melhorar aspectos de escrita dos mesmos com verificadores semânticos e sintácticos. Ambas as DSLs incorporaram aspectos de modelos cognitivos na esperança de aumentar a compreensão de todos os *stakeholders*.

[1] CSS - Cascading Style Sheet - Linguagem para especificar formatação da uma página HTML

4.4.1 Características

Existem dois tipos de linguagens, LPG[2] e DSL. A diferença entre elas recai sobre o espectro de aplicabilidade de cada uma. As LPG são desenhadas para programadores e podem ser aplicadas a qualquer domínio e não trazem nenhuma vantagem em domínios específicos. No entanto, uma DSL é construída com um certo domínio em mente e tira proveito de ferramentas próprias para mapear esses conceitos com uma alta abstracção de forma a minimizar os erros e optimizar o desenvolvimento. Independentemente do grupo a que pertença, qualquer linguagem possui [Voe+13]:

1. **Sintaxe concreta**: notação na qual o utilizador interage com a linguagem.
2. **Sintaxe abstracta**: estrutura de dados que representa a semântica relevante de um programa. É utilizada para análise e processamento de programas.
3. **Semântica estática**: restrições que o programa tem de obedecer para além das restrições da sintaxe abstracta.
4. **Semântica de execução**: refere-se ao significado do programa quando está a ser executado.

Para além da definição sintaxe e semântica, é necessário ter um *engine* de execução para executar a linguagem. Existem duas formas de fazê-lo: por geração ou interpretação. O método de interpretação traduz o programa da DSL para um *engine* de uma plataforma (de uma GPL) já existente. No entanto, por geração é necessário criar um *engine* em cima de uma plataforma alvo.

A Figura 4.3 mostra o ciclo de desenvolvimento de uma DSL. O desenvolvimento começa com o reconhecimento do domínio. Conhecendo o domínio, através de especialistas do mesmo, ou utilizando modelos de *features*[3], simplifica-se a criação de um meta-modelo. Esse meta-modelo corresponde à sintaxe abstrata da linguagem. A partir do meta-modelo, expressamos quais são as relações entre os vários componentes do domínio. Após isso, construímos a sintaxe concreta, adicionando mais restrições de domínio e símbolos a cada elemento. De seguida, chega a parte de implementação, que pode ser através de um gerador ou interpretador. Com a linguagem pronta, são feitos testes com utilizadores para verificar se correspondem às suas necessidades e se são necessárias melhorias. Caso corresponda, a DSL está pronta a ser utilizada.

Ao concluir este processo, obtemos uma DSL e os seus benefícios inerentes. Em primeiro lugar, um aumento de produtividade, visto que, a própria criação da DSL veio melhorar a usabilidade anterior vinda da GPL e automatizar a transformação de código para a plataforma final. Outro aspecto importante é um aumento de qualidade. Existe uma probabilidade menor de cometer erros, devido às transformações automáticas para o código da plataforma alvo desejada e na própria DSL existem restrições próprias na sua utilização derivadas do seu domínio. Sendo uma ferramenta de domínio específico,

[2]LPG - Linguagens de Propósito Geral, em inglês (General Porpose Languages- GPL): (e.g. Java, C++)
[3]Modelo de *features* - Modelo que expressa variabilidade de um sistema

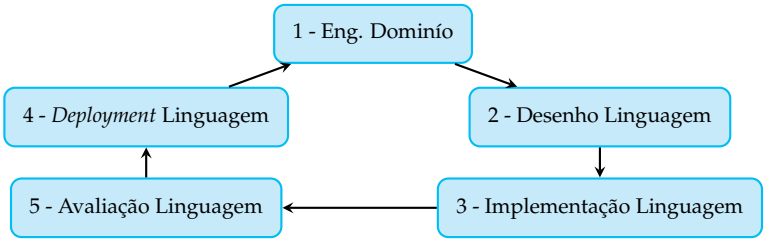


Figura 4.3: Ciclo de Devensolvimento de uma DSL baseado em [BAG12]

serve também como ferramenta de comunicação dentro do próprio domínio, visto que pode ter uma sintaxe concreta própria das entidades do domínio e facilitar a partilha de ideias e problemas.

Apesar destas vantagens, existe o reverso da medalha. Construir uma DSL requer tempo e exige esforço, logo nem todas as empresas estão dispostas a fazê-lo, dependendo de um estudo custo-benefício. Com isso, também se levanta a questão da educação dos utilizadores do domínio e o custo de manter e evoluir a linguagem com o ritmo e necessidades do domínio. A justificação de construir uma DSL está intimamente ligada ao benefício ganho pela mesma, dependendo muito do domínio de aplicação.

4.4.2 Ferramentas de apoio à construção

De forma a construir linguagens especificas de domínio abstraindo-se de detalhes de implementação do *engine* da linguagem, existem *frameworks* que permitem iniciar o desenvolvimento da DSL partindo com um modelo do domínio da solução. A lista seguinte apresenta algumas **frameworks** disponíveis:

1. **EuGENia GMF**: um *framework* que gera automaticamente artefactos para o *Grapical Modeling Framework* do *Eclipse*. É uma extensão que veio facilitar bastante a criação de DSLs onde para definir uma linguagem basta definir via um ficheiro *Enfatic* ou produzindo um com recurso do *Ecore*.
2. **Xtext**: faz parte do ecossistema do *Eclipse* e serve para criar DSL textuais.
3. **JetBrains MPS**: o MPS significa *Meta Programming System* e tem como objetivo principal estender linguagens. A ferramenta oferece uma linguagem base que é estendida de forma a criar uma DSL. Utiliza uma representação não textual do código do programa utilizando uma AST[4]. Com essa abordagem consegue fazer verificação em tempo real de erros de sintaxe e elimina a necessidade de ter *parsers*.

As linguagens apresentadas nesta dissertação representam cenários *Behavior-Driven Development* e modelos de domínio e foram implementadas com a ferramenta *EuGeNia*. Ambas são linguagens gráficas com uma semântica bem definida, com o objetivo de melhorar aspectos relativos à escrita de cenários BDD, a sua transformação para casos de

[4]Abstract Syntax Tree - árvore com propriedades dos nós afim de representar o código do programa

testes, como também a comunicação, com recurso a mapas mentais, entre a equipa de desenvolvimento e os *stakeholders*.

4.5 Resumo

Neste capítulo foram introduzidos os conceitos de MDD e de DSL. A definição de um modelo e como as transformações feitas entre eles são executadas foram abordadas de forma a ter uma compreensão de como isso se encaixa na construção de uma DSL. No próximo capítulo vamos apresentar o trabalho relacionado para averiguar a adoção do BDD, quais os modelos comportamentais utilizados em processos ágeis e abordagens de requisitos centrados em utilizadores.

5

Trabalho Relacionado

Esta dissertação, tem como propósito, utilizar a prática BDD para capturar e validar requisitos funcionais. O BDD expressa esses requisitos sob a forma de cenários comportamentais. De forma a avaliar a adoção do BDD na comunidade científica e na indústria e investigar quais as alternativas de modelos comportamentais utilizados em processos ágeis, realizámos uma revisão da literatura. Além da revisão da literatura sobre o BDD foram também reportados trabalhos relacionados com abordagens de requisitos centrados nos utilizadores. Este capítulo tem o propósito de expor ao leitor a revisão da literatura realizada e os trabalhos relacionados encontrados.

5.1 Método

A revisão da literatura efetuada está agrupada em três fases distintas: planeamento, execução e demonstração de resultados [KC07]. Na fase de planeamento o objetivo é desenvolver um protocolo que especifica como é que a revisão vai ser conduzida, quais são as questões de pesquisa, as palavras de buscas e regras de inclusão e exclusão. A segunda fase é de pesquisa, onde o protocolo definido na fase de planeamento é executado. A terceira e última fase é a exposição dos resultados encontrados.

5.1.1 Questões de pesquisa

De forma a tentar descobrir quais são os modelos comportamentais utilizados e a adoção do BDD em processos ágeis, as perguntas de pesquisa desenhadas foram as seguintes:

1. **Questão de pesquisa 1**: quais são os modelos comportamentais utilizados nos processos ágeis?

2. **Questão de pesquisa 2**: qual é a adoção do BDD nos processos ágeis?

De acordo com [KC07] as questões de pesquisa podem ser divididas em quatro aspectos: *população*, *intervenção*, *comparação* (caso se aplique) e *resultados*. A estrutura utilizada nas questões de pesquisa foi a seguinte:

1. **População:** profissionais envolvidos em processos (metodologias) ágeis
2. **Intervenção:** ferramentas de BDD ou modelos comportamentais utilizados no desenvolvimento
3. **Resultados:** o objetivo deste estudo é descobrir quais são os modelos comportamentais utilizados em processos ágeis de forma a saber que características podem ser aplicadas no modelo que queremos criar e se de facto o BDD tem vindo a ser investigado pela comunidade e a indústria.

5.1.2 Estratégia de pesquisa

De forma a agregar dados para obter respostas às nossas questões, foi feito uma extração de *palavras-chave* de pesquisa. Essas foram depois inseridas em motores de busca de bibliotecas de artigos científicos. A Tabela 5.1 sumariza as bibliotecas utilizadas e a Tabela 5.2 mostra as *palavras-chave* utilizadas para as questões de pesquisa *1* e *2*. Em relação à segunda questão, também foi utilizado o *Google* para a procura de ferramentas de suporte ao BDD de forma ter uma perceção da sua adoção na indústria.

Tabela 5.1: Bibliotecas utilizadas para a realização da pesquisa

Bibliotecas
ACM Digital Library
IEEE Computer Society Digital Library
Science@Direct

Tabela 5.2: Palavras-chave utilizadas na pesquisa para a questão 1 e 2

Questão 1	**Questão 2**
Agile **AND** "Requirements Engineering" **AND** "Modeling"	"Behaviour-Driven Development" **OR** "Behavior-Driven Development"

5.1.3 Critérios de inclusão e exclusão

Durante a pesquisa de artigos e livros, os critérios de seleção foram os seguintes: algum artigo ou livro que mencionasse as *palavras-chave* da pesquisa e tivesse sido publicado seria aberto e o resumo lido. Caso o resumo fosse de acordo com as intenções das perguntas, era guardado para futura leitura. Especificamente para a *questão 1* as regras de inclusão foram:

1. Artigos com casos práticos de desenvolvimento ágil;

2. Artigos com estudos sobre documentação ágil;
3. Artigos de modelação ágil.

Relativamente à *questão 2* foram:

1. Todos os artigos (livros ou dissertações) que fizessem estudos que incluem-se o BDD (expandindo-o ou integrando com outras práticas);
2. *Surveys* de práticas de testes;
3. Artigos após 2006 (ano da criação do BDD).

5.2 Resultados obtidos

As secções seguintes vão apresentar os resultados obtidos para as questões de pesquisas definidas nesta revisão da literatura.

5.2.1 Quais são os modelos comportamentais utilizados em processos ágeis?

Os resultados obtidos para a *questão 1* estão apresentados na Tabela 5.3. No caso das pesquisas feitas no *Springer Link*, os resultados foram filtrados por: *Computer Science*; *SWE* e *Requirements Engineering*. Nas pesquisas realizadas no *Science Direct* foram limitados aos tópicos: *requirements engineering* e *requirements elicitation*. No caso da *ACM Digital Library* as pesquisas foram limitadas por os *sponsors ACM* e *IEEE* e com o valor *yes* no campo *AbstractFlag*.

Tabela 5.3: Artigos encontrados para a questão 1

Base de dados	Artigos Encontrados	Artigos Selecionados	Artigos Revistos
IEEE Digital Library	16	5	1
Science Direct	8	3	2
Springer Link	36	8	3
ACM Digital Library	26	8	2
Total	**86**	**24**	**8**

Também foi feita uma pesquisa no *Google Scholar* e foram adicionados mais 4 artigos, ficando 12 artigos revistos no total. Os trabalhos encontrados são detalhados em seguida.

Paetsh e *Lucia* [PEM03; LQ10] fazem uma comparação da abordagem de RE tradicionais com o Desenvolvimento Ágil. Ambos identificam as seguintes práticas ágeis para captura de requisitos: envolvimento do cliente, em todo o processo de desenvolvimento, entrevistas e sessões JAD[1]. Já em termos de validação falam no uso de protótipos, testes de aceitação e reuniões de revisão.

Em termos de modelação de *design*, *Paetsh* diz que os modelos criados tem um propósito de comunicação e não de documentação. A escolha do modelo e o seu nível de

[1]Joint Application Development (JAD) - Integrar o cliente com a equipa de desenvolvimento durante a produção de *software*

abstração dependem do contexto em que é utilizado. *Rubin* [RR11] menciona três categorias de modelos comportamentais: *Control Flow* (e.g. Diagrama de actividades), *Data Flow* (e.g. DFDs) e *State Machines* (e.g. Digrama de estados).

O trabalho de *Rubin* [RR11] descreve uma abordagem para criar documentação em processos ágeis com base no conhecimento de domínio. Propõe um modelo composto por elementos de modelação orientada ao domínio (*Problem Frames*), a dados (*Entity-Relationship*), ao comportamento (*State-Machines*) e aos objetivos (*KAOS*). A partir desse modelo, a que ele designa de *REDK* (*Requirements Engineering Domain Knowledge*), são instanciados os requisitos do problema. A abordagem consiste em quatro componentes:

1. ***Conceptual Subsystem***: um subsistema que representa o *REDK* a fim de ser instanciado com os objetos do sistema.
2. ***Processing Subsystem***: um conjunto de subsistemas que implementa todos os assuntos relacionados com as estruturas de dados, I/O, interações e interfaces de utilizador
3. ***Instance Base***: um subsistema que vai representar os requisitos do sistema baseado no *Conceptual Subsystem*.
4. ***Mediator Component***: encarregue das interações entre o *Instance Base* e o *Processing Subsystem*.

O trabalho de *Soares* [SVV11] descreve uma abordagem para modelar requisitos através de *sysML*. A abordagem consiste em modelar os requisitos em modelos *Use Case* e Requisitos *sysML*. Também refere uma forma de visualizar os requisitos em tabelas, ordenados por prioridade. Os diagramas de requisitos em *sysML* tem a vantagem de poderem ser extensíveis e criar um linguagem de domínio própria.

Michal [ŚNS05] propõe a construção de cenários *SVO* (*Subject*, *Verb*, *Objects*). Um cenário *SVO* começa sempre com o sujeito, é seguido por um verbo que designa a acção e um ou mais objetos que são nomes. *Michal* estende o UML para criar cenários de *Use Cases SVO* e constrói uma ferramenta de suporte. Essa ferramenta tem a capacidade de sugerir nomes na construção de cenários e construção automática de modelos de domínio.

O trabalho *Winget* [WS11] recai sobre o desenvolvimento de jogos. Tenta descobrir qual a documentação utilizada no desenvolvimento de jogos. Ele reporta que as práticas ágeis de prototipagem e desenvolvimento iterativo são abordagens utilizadas na indústria de jogos.

Foram também encontrados casos de estudos em áreas relacionadas com a saúde [TFS12; STJ11; GK06], utilizando um processo de desenvolvimento ágil, com colaboração activa dos *stakeholders* e utilizando técnicas de UCD[2]. As técnicas utilizadas para captura de requisitos foram: análise de tarefas, entrevistas, comunicação directa, sessões JAD e diagramas *Use Case*. O uso de protótipos é a escolha principal para a validação de requisitos. A prática de *análise de tarefas* consiste em compreender todos os passos

[2]UCD - User-Centered Design

necessários para a realização de uma tarefa. São utilizados diagramas constituídos com as subtarefas necessárias para a realização de uma tarefa. O primeiro elemento do diagrama corresponde à tarefa a realizar e essa é decomposta em subtarefas, onde, a cada uma, é atribuído um número que corresponde à ordem de execução a fim de atingir a tarefa principal.

Relativamente a casos industriais, *Cao* [CR08] fez uma estudo de práticas ágeis de Engenharia de Requisitos a 16 empresas e concluiu que a comunicação directa, com recurso a *User Stories* e o uso de protótipos são as mais adoptadas. Já *Maya* [Dan+13], na sua revisão sistemática da literatura sobre priorização de requisitos em processos ágeis de larga escala, descobre um novo tipo de artefacto ágil, *Delivery Story*. Segundo *Maya*, uma *Delivery Story* é uma extensão de *User Story* com especificações funcionais mais detalhadas. A partir dessa especificações são tomadas decisões de *design* e criados cenários de testes, por parte da equipa de *design* e de testes, respectivamente.

Ao analisar os artigos selecionados, outro rumo de pesquisa nasceu, com o artigo de *Memmel* [MGR07], onde foi feita uma comparação de processos ágeis com *HCI*[3]. Da comparação é chegado à conclusão que ambas as disciplinas utilizam os modelos como: *Use Cases*, protótipos e cenários. Isso fez-nos pesquisar nessa direção. Encontrámos uma revisão sistemática [Sil+11], onde foram reunidos 58 artigos, de 2000 a 2010, relacionando métodos ágeis com UCD. Ele encontrou 21 artigos que mencionam o uso de protótipos como boa prática de requisitos e usabilidade e 17 outros artigos a mencionar o uso de *User Stories*.

Além dos resultados encontrados, durante a pesquisa sobre modelação ágil, na parte introdutória da dissertação, foram encontradas referências a considerar para ajudar a responder a esta questão. *Ambler* [Amb02], ao falar de modelação ágil diz que devemos utilizar um modelo com o intuito de perceber um problema. Esse modelo deve facilitar a comunicação, ter um propósito e não devem existir restrições a qual modelo utilizar. Dependendo do contexto da situação, quantos mais soubermos melhor. *Cockburn* e *Hunt* [Coc06; Hun06] partilham da opinião de *Ambler*, afirmando que um modelo deve satisfazer um objetivo e nada mais, desde que consiga transmitir informação suficiente para a próxima pessoa continuar o seu trabalho, está completo. Na revisão sistemática da literatura sobre artefactos ágeis de *Grober* [Grö13], foram analisados 76 artigos e foi concluído que em relação a artefactos de *design*, os processos ágeis não são específicos. As *User Stories* são mencionadas como uma visão do sistema e o código é a realização dessa visão, sem existir algo entre esses dois extremos.

Leffingwell e *Copllien* [Lef11; CB10], nos seus livros, abordam também o tema, mencionando o uso de modelos para ajudar a partilha de informação e compreensão em projetos grandes com desenvolvimento ágeis. *Leffingwell* diz que, como os processos ágeis têm vindo a ganhar popularidade e são aplicados em projetos complexos, com um número elevado de membros, nasce a necessidade de utilizar modelos comportamentais

[3]HCI - Humam Computer Interaction

para partilhar informação. Indica, também, certos modelos comportamentais que podem ser utilizados: Diagramas de Atividades; Sequência; Estado; Use Cases; *User Stories*; Protótipos; DFDs; *Behavior-Driven Development*.

5.2.2 Qual é a adoção do BDD nos processos ágeis?

Esta questão tem como objetivo tentar descobrir qual é a adoção do *Behavior-Driven Development* nos processos ágeis. Após uma pesquisa inicial, não foi encontrado nenhum artigo que comprove se o BDD integrou o processo ágil. Foi apenas encontrado um artigo *survey* de práticas de testes no Canadá que menciona o BDD. Dado isso, foi tomada a decisão de fazer uma pesquisa para verificar se a comunidade científica e a indústria têm vindo a trabalhar para explorar o BDD.

A pesquisa foi elaborada com o propósito de encontrar livros, dissertações e artigos que estudam o BDD. Relativamente à adoção da indústria a pesquisa foi direcionada para a procura de ferramentas que suportam o BDD. Os artigos encontrados estão apresentados na Tabela 5.4. No caso das pesquisa feitas no *Springer Link*, os resultados foram filtrados por: *Computer Science*; *SWE* e do ano 2006 a 2014.

Tabela 5.4: Artigos encontrados para a questão 2

Base de dados	Artigos Encontrados	Artigos Utilizados
IEEE Digital Library	11	4
Science Direct	5	0
Springer Link	25	3
ACM Digital Library	64	11
Total	**105**	**18**

Como o resultado de artigos utilizados referentes as bibliotecas foi considerado pouco, foi também feita uma procura no *Google Scholar* com a mesma palavra-chave e foram adicionados mais 15 artigos, perfazendo 33 artigos no total. No que diz respeito a ferramentas de suporte ao BDD, foi feita uma pesquisa no *Google* com: *BDD tools* e *Behavior Driven Deevelopment Tools*. A distribuição dos resultados é apresentado na Figura 5.1.

Em relação aos 33 artigos utilizados relacionados com BDD: 6 deles dissertações de mestrado, 5 livros e 22 artigos científicos. Os livros são sobre *frameworks* que implementam o BDD e os artigos diferem desde desenvolvimento de aplicações na saúde, *cloud* a circuitos digitais. Os artigos selecionados são apresentados no Anexo A. O único *survey* [GZ13] onde o BDD é referenciado é sobre um estudo de práticas de testes no Canadá durante o ano 2010. Os seus resultados mostram uma adoção de **18%** num grupo de 246 participantes.

No que diz respeito às ferramentas encontradas, **19** ao todo, onde se destacam algumas por serem construídas pela *Microsoft* e pela *Pivotal Labs*. As outras são maioritariamente *open-source*. Destacam-se duas ferramentas que já tem livros de suporte, o *rspeck*, e o *cocumber*. O *JBehave*, foi construída pelo criador do BDD, *Dan North*, e mantêm-se em desenvolvimento ativo desde então, sendo das mais completas do mercado. A Figura 5.2 mostra o número de ferramentas BDD disponíveis desde 2006 a 2013.

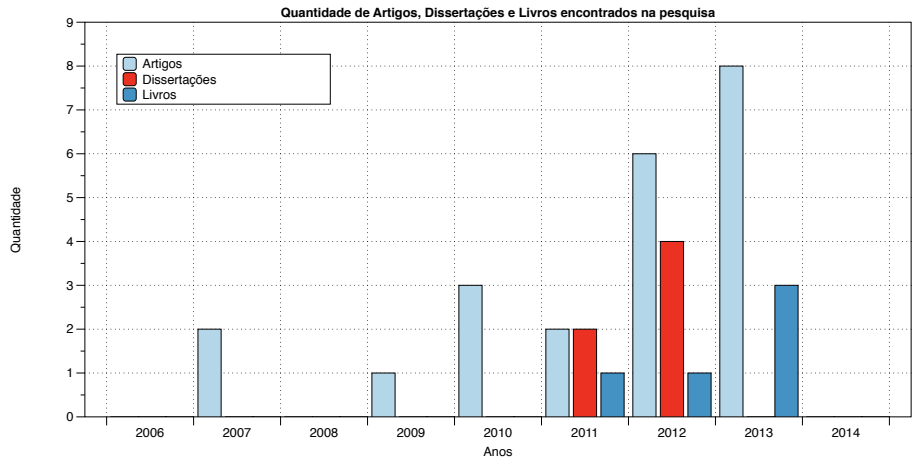


Figura 5.1: Gráfico do número de artigos, dissertações e livros sobre o BDD publicados desde 2006

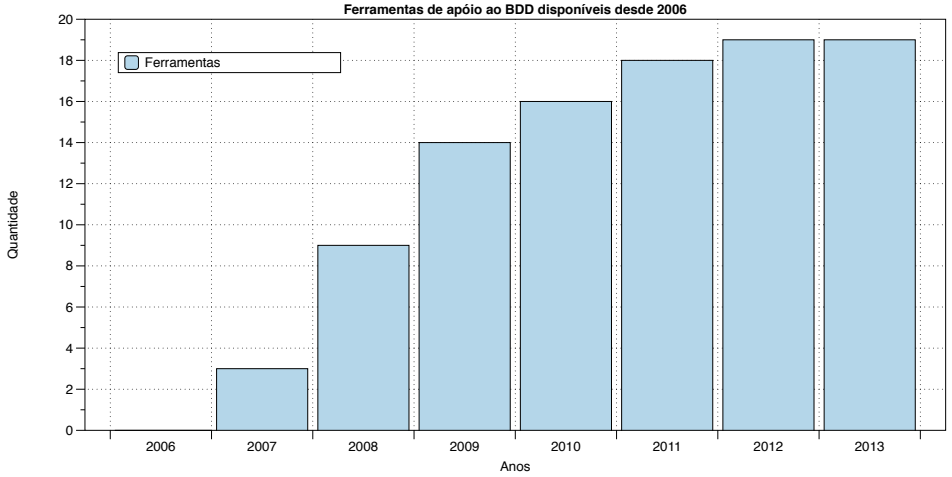


Figura 5.2: Gráfico do número de ferramentas de apoio ao BDD disponíveis desde 2006

5.3 Abordagens de requisitos centrados no utilizador

Os trabalhos relacionados com abordagens centradas no utilizador, mencionam a utilização de exemplos para o desenvolvimento de meta-modelos ou dar a possibilidade aos *stakeholders* de criarem a sua própria notação. Os trabalhos são detalhados em seguida.

Os trabalhos de *Lopez* e *Cuadrado* [Lóp+13; CLG12] exploram uma abordagem de criação de meta-modelos baseada em exemplos. Os meta-modelos são criados iterativamente e interativamente através de exemplos criados pelos clientes. Os clientes criam exemplos de casos concretos da sua linguagem. Esses exemplos são chamados fragmentos. A partir dos fragmentos criados, um engenheiro constrói um meta-modelo base. Esse, é depois validado com um outro conjunto de exemplos criados por clientes com auxílio de engenheiros. Após uma validação sucedida, o meta-modelo é transformado para meta-modelos para implementação, por exemplo no *EMF*.

O trabalho de *Bak* [Bak+13] recai também sobre uma abordagem de criação de modelos através de abstrações e exemplos. É um artigo visionário, onde é argumentado que construir modelos com base em exemplos concretos melhora a sua qualidade. São mencionadas técnicas como o BDD que promovem a participação dos *stakeholders* com a utilização duma linguagem ubíqua e cenários com exemplos concretos para a criação de testes de aceitação, no entanto, realçam que essas técnicas não são comuns para a construção de modelos. É argumentado também que criando modelos utilizando exemplos concretos podem melhorar a compreensão dos *stakeholders* envolvidos.

Já o trabalho de *Wuest* [WG11] aborda a possibilidade de criar notações desenhadas pelos clientes. Uma modelação de requisitos baseados em *sketchs*. A ideia principal é que sejam os próprios clientes a especificarem uma notação para especificar requisitos. A partir da notação que os clientes criam são construídos modelos personalizados. Os modelos personalizados são posteriormente transformados para modelos clássicos para que a equipa de desenvolvimento consiga analisá-los em detalhe. Este trabalho tem uma ferramenta promissora da abordagem, *FlexiStetch*, apresentada no *link*[4].

5.4 Discussão

A revisão da literatura sobre modelos comportamentais ágeis, relevou que parece não existir regras sobre quais os modelos a utilizar no processo de desenvolvimento. Cabe ao profissional, no seu conhecimento decidir qual o modelo a utilizar. No entanto, parece existir uma tendência de algumas práticas de usabilidade, e.g., a prototipagem como prática regular de validação de requisitos.

Relativamente à adoção do BDD, os dados não explicitam concretamente se é utilizado em processos ágeis, e seria muito difícil conseguir determinar isso sem um estudo

[4] http://www.ifi.uzh.ch/rerg/research/flexiblemodeling/flexisketch.html

intensivo com um vasto número de empresas. No entanto, com este número de aplicações desenvolvidas por diversas entidades, os livros existentes, as dissertações de mestrado e o número de artigos, existe uma evidência que a comunidade está a interessar-se pelo BDD.

Em relação às abordagens centradas nos utilizadores, os trabalhos mencionados mostram uma aposta em partir de exemplos para a definição de modelos e a capacidade de deixar os utilizadores criarem as suas próprias notações. De notar que nenhuma das abordagens mostradas mostrou validação em relação às notações utilizadas, como sendo mais facilmente compreensíveis pelos *stakeholders*.

5.5 Resumo

Neste capítulo foi realizada uma revisão da literatura sobre modelos comportamentais ágeis e a adoção do BDD. A revisão indica que existem vários modelos comportamentais aptos para processos ágeis, mas cabe ao profissional escolher quais se adaptam melhor à situação. Em relação à adoção do BDD, o número de artigos relacionados têm vindo a aumentar desde o seu aparecimento. Finalmente, sobre as abordagens de requisitos centrados em utilizadores, os trabalhos sugerem uma participação ativa dos *stakeholders* com a contribuição de exemplos para desenvolver linguagens ou a definição da própria notação a utilizar nos modelos.

6

A Abordagem SnapMind

Este capítulo apresenta ao leitor o *framework* criado nesta dissertação e a sua integração no processo de desenvolvimento do BDD. O *framework* proposto nesta dissertação é composto por duas linguagens de modelação, uma para modelos de domínio e outra para cenários comportamentais BDD. As linguagens são inseridas no processo de desenvolvimento BDD (descrito na secção 6.1). As linguagens são baseadas em modelos cognitivos e interagem entre si, onde a informação dos modelos de domínio é incorporada na criação de cenários comportamentais BDD.

6.1 Processo

O processo no qual vão ser inseridas as linguagens desenvolvidas nesta dissertação tem como base o processo de desenvolvimento de *software* do BDD. A Figura 6.1 mostra o processo resultante. O processo está dividido em três fases: planeamento, análise e implementação. Os retângulos significam as atividades a realizar. As atividades a verde são as tarefas adicionadas. As atividades assinaladas com uma estrela têm suporte duma DSL de apoio à execução.

O processo inicia-se quando o(s) *stakeholder*(s) e a equipa de desenvolvimento pretendem construir um *software* para resolver um problema existente (atividade 1). Dessa intenção de resolver um problema, nascem várias funcionalidades que os *stakeholders* desejam implementadas (atividade 2). De forma a capturar as entidades presentes no domínio do problema, é criado um modelo de domínio, atividade proposta nesta dissertação (atividade 3). Cada uma dessas funcionalidades é especificada em *User Stories* para descrever com mais detalhe a sua função (atividade 4). Das *User Stories* criadas, são

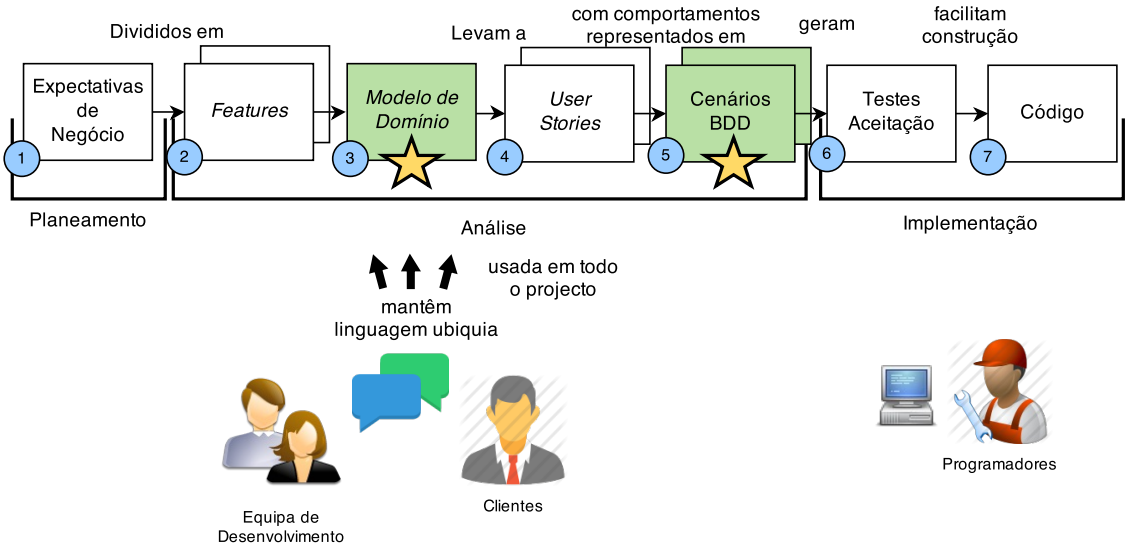


Figura 6.1: Processo BDD proposto

produzidos cenários comportamentais BDD de forma a detalhar o comportamento desejado (atividade 5). A ordem de criação de cenários comportamentais é feita de acordo com uma priorização feita pelos *stakeholders* das *User Stories*. Desses cenários BDD, são criados testes de aceitação de forma a verificar se as necessidades são corretamente implementadas (atividade 6). A última atividade (atividade 7) diz respeito à construção do código para satisfazer os testes de aceitação e construir as funcionalidades previamente planeadas.

As atividades adicionadas (assinaladas a verde na Figura 6.1) servem de incorporação das linguagens criadas nesta dissertação. A primeira atividade, atividade 3, serve para capturar informação do domínio sob a forma de um mapa mental. A utilização de um mapa mental é devido à premissa desta dissertação, sobre o uso de modelos com notações cognitivas adaptadas ao domínio, para facilitar a participação dos *stakeholders* no processo de desenvolvimento.

O seu processo de utilização é apresentado na Figura 6.2. Nesta atividade, a equipa de desenvolvimento, os *stakeholders* e um *designer* reúnem-se e criam um modelo de domínio com a informação do *software* a desenvolver. A Tarefa 1, *Construir Modelo Conceptual*, serve para que o especialista de domínio explique ao engenheiro de requisitos quais são as entidades e os respetivos atributos que constituem o domínio que a aplicação vai tratar. A cada entidade criada é lhe atribuída uma notação visual especifica. Essa notação vai ser construída com o auxílio dum *designer* e do especialista de domínio. A notação vai corresponder a um ícone que pode ser inicialmente procurado num repositório[1] para verificar qual agrada mais ao especialista de domínio. Após o especialista de domínio considerar que o domínio está mapeado para prosseguir ao desenvolvimento, começam as próximas tarefas. O engenheiro de requisitos procede a analisar o modelo de domínio

[1]Repositório - https://www.iconfinder.com

e a refiná-lo (Tarefa 1.1), verificando se é benéfica a introdução de entidades associativas ou a inserção de enumerados.

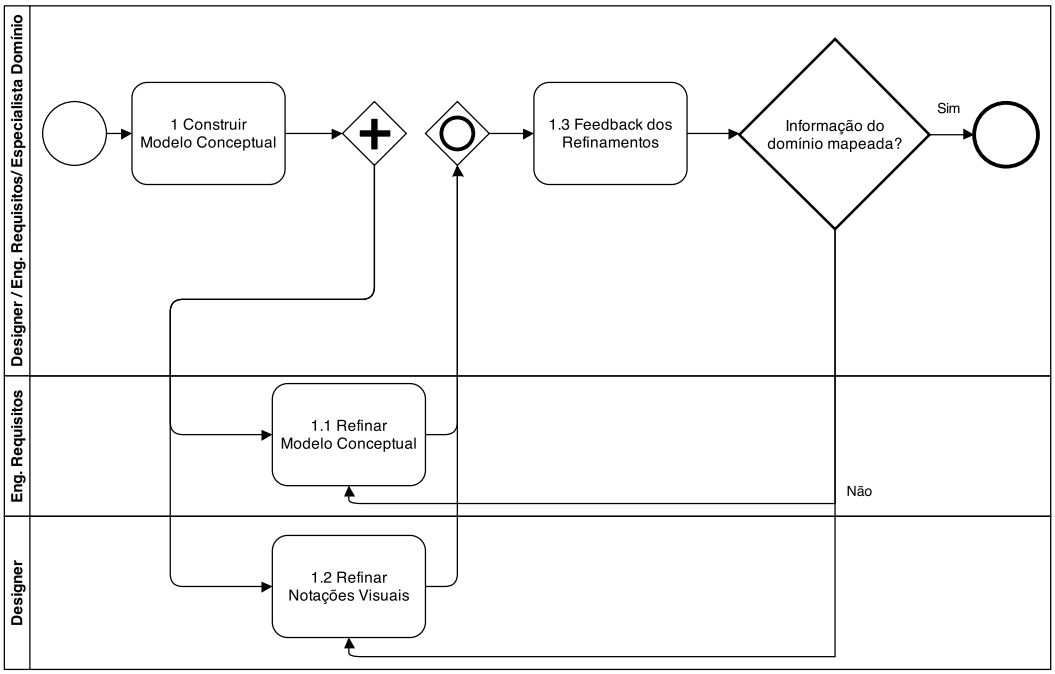


Figura 6.2: Processo da actividade 3 - Modelo de Domínio

Simultaneamente, o *designer* visualiza todos as notações visuais escolhidas para as entidades e verifica se as mesmas partilham o mesmo estilo de desenho, tamanho, perspetiva, para garantir que não existam discrepância visuais significativas entre as mesmas [App14; And14] (Tarefa 1.2). Caso não tenham sido encontrados nenhuns exemplos aprovados pelo especialista de domínio em relação a alguma entidade, são desenhadas várias notações distintas para serem discutidas na próxima reunião.

Ao terminar os refinamentos, existe uma reunião com o especialista de domínio para mostrar o modelo de domínio refinado e aguardar a sua aprovação (Tarefa 1.3). Caso a validação seja positiva, o processo segue. Caso contrário, são apontadas as modificações a realizar. Os refinamentos são feitos novamente até existir um consenso entre os três elementos.

A outra alteração no processo recai no uso da linguagem utilizada na atividade 5. Nesta atividade é pretendido o uso de uma ferramenta que incorpora uma linguagem específica de domínio para especificar comportamento sob a sintaxe do BDD. Essa ferramenta recebe o modelo de domínio previamente criado, para auxiliar na construção dos cenários comportamentais. O processo da atividade está demonstrado na Figura 6.3. É escolhida uma *User Story* para escrever um ou mais cenários comportamentais. Qualquer desenho dum cenário comportamental é construído com a presença de um especialista de domínio para garantir a especificação correta. Os cenários podem ser escritos textualmente também, de forma a ter rascunhos e a estruturar a melhor forma de como o

comportamento é executado, no entanto, o cenário depois é traduzido para cenário gráfico. As vantagens de ter o cenário graficamente são: o suporte de uma DSL com regras de verificação, garantir que está de acordo com o modelo conceptual previamente definido e a criação dos testes de aceitação automaticamente. Caso o cenário não consiga ser especificado graficamente devido a não ter todas as entidades mencionadas específicas, é necessário modificar o modelo conceptual e voltar a iniciar a atividade 3 do processo BDD. Caso contrario, o processo segue para a atividade 6.

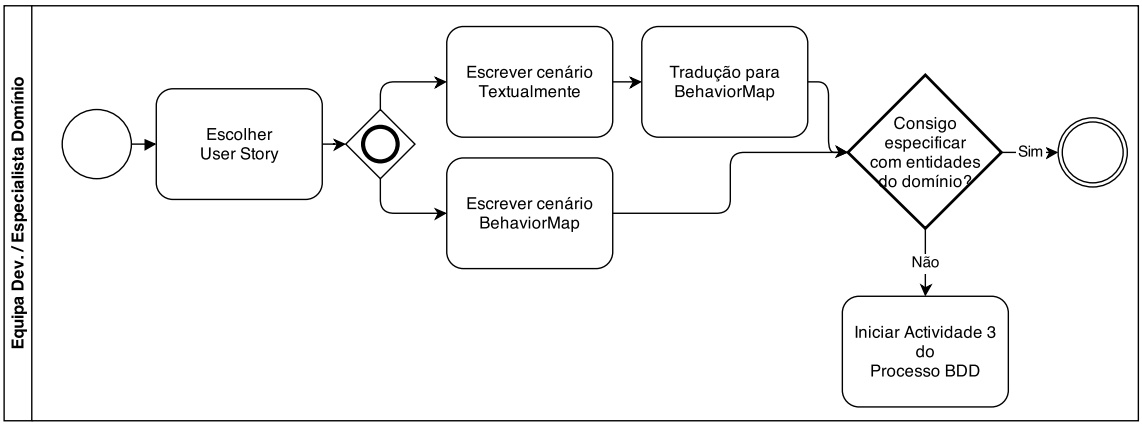


Figura 6.3: Processo da actividade 5 - Cenários BDD

Com as ferramentas introduzidas existe suporte para as atividades 6 e 7 do processo. A ferramenta para construir cenários BDD gráficos tem a capacidade de gerar testes *JUnit* automaticamente e a ferramenta para construir os modelos conceptuais consegue também transformar o modelo conceptual criado num conjunto de classes *Java*. As transformações serão explicadas em mais detalhe nas secções 6.2.1 e 6.2.2.

6.2 Linguagens desenvolvidas

Esta dissertação focou-se em construir duas linguagens de apoio ao desenvolvimento ágil, com foco na compreensão de utilizadores (*stakeholders*) não especialistas no desenvolvimento de *software*, de modo a tentar melhorar a sua participação no processo de desenvolvimento. Para tentar alcançar esse objetivo, os modelos utilizados nas nossas linguagens são baseados em mapas mentais onde foi adotada a sua sintaxe concreta. Ambas as linguagens desenvolvidas têm características comuns como as seguintes apresentadas.

Características Cognitivas

Uma das vantagens das linguagens desenvolvidas é a capacidade de adaptar a sua sintaxe concreta de acordo com o domínio em que são aplicadas. De acordo com [Fow10], o sucesso de uma linguagem gráfica depende da correspondência das suas notações visuais com o domínio para qual é aplicado. As linguagens desenvolvidas tiveram isso

em consideração e têm a capacidade de definir as suas notações visuais com colaboração de utilizadores (design participativo [Cai+13; Bød04]). Através desta capacidade, permitimos a participação de especialistas de domínio no processo de escolha de notações visuais que vão acompanhar o desenvolvimento do projeto. Isto vai de acordo com a teoria da Física das Notações [Moo09], em que é afirmado que deve existir uma correspondência de 1:1 entre conceitos e símbolos adotados, i.e. deve ser possível distinguir vários símbolos e existir transparência semântica entre todos.

Além da sintaxe adaptada ao domínio, as linguagens incorporam as características de mapas mentais: organização de ideias e conceitos, ênfase em palavras relevantes, associação entre elementos e arcos, utilização de cor por cada ramo, grupos de ideias e suporte à memória visual e criatividade.

Modificações ao EuGENia

De forma a enriquecer as ferramentas construídas para terem a possibilidade de ter sintaxes visuais personalizadas, adição de *pop-ups*, algoritmo de reordenação, entre outras funcionalidades, foi necessário estender o EuGENia. A criação de *plugins* no EuGENia é feita em duas partes: a adição de uma entrada num ficheiro *xml* e o código *Java* com o respetivo comportamento desejado. As adições criadas para ambas as ferramentas foram as seguintes:

1. Capacidade de alterar notação visual de nós;
2. Sincronização de nós que representam o mesmo conceito ao serem modificações;
3. Algoritmos de reordenação semelhante ao dum mapa mental;
4. Transformação *in-place* para outros artefactos.

6.2.1 DomainMap - Linguagem para Modelar Domínios

A linguagem para modelar domínios, *DomainMap*, tem o intuito de auxiliar as duas frentes de desenvolvimento de *software*, *stakeholders* e equipa de desenvolvimento, a agrupar informação de domínio necessária para o desenvolvimento de *software*. A atividade de criar o modelo de domínio, atividade 3, introduzida no processo de desenvolvimento BDD (apresentado na Figura 6.1), consiste em capturar e analisar os elementos principais do domínio dum sistema.

O *DomainMap* não é a linguagem principal desta dissertação, mas foi desenvolvida para criar um *framework* mais abrangente com a premissa de que os mapas mentais garantiam a proximidade dos *stakeholders* no processo de desenvolvimento graças às suas características cognitivas. Assim, o *DomainMap* é utilizado para auxiliar a escrita de cenários BDD, incorporando a informação do domínio, na segunda e principal linguagem desenvolvida da dissertação, o *BehaviorMap* (secção 6.2.2).

Um exemplo de um modelo de domínio produzido pela ferramenta é apresentado na Figura 6.4. O exemplo representa o domínio de uma aplicação para a empresa *Mobciti*. A *Mobciti* é uma *start-up* Brasileira com foco em difundir publicidade em transportes públicos. O sistema representado na Figura 6.4, *Audio Bus*, é o sistema principal da *Mobciti* e o seu principal objetivo é transmitir publicidade via áudio em autocarros de acordo com o contexto de localização ou critérios de clientes. O *Audio Bus* é constituído por seis entidades: *Bus*, *Coordinates*, *Broadcast*, *MobileDevice*, *Route* e *BusStop* e três enumerados: *BroadcastType*, *Status* e *ContentType*. Cada entidade é identificada pelo seu nome e notação visual. Uma entidade é caracterizada pelos seus atributos. Por exemplo, a entidade *Bus* é caracterizada por ter os atributos: *licensePlate*, *brand* e a entidade *MobileDevice*. A entidade *Route* é uma entidade associativa, semelhante a uma classe associativa, que existe duma relação entre duas outras entidades: *Bus* e *BusStop*. O ícone de uma ferramenta no canto inferior direito da uma entidade é indicativo de restrições impostas na mesma.

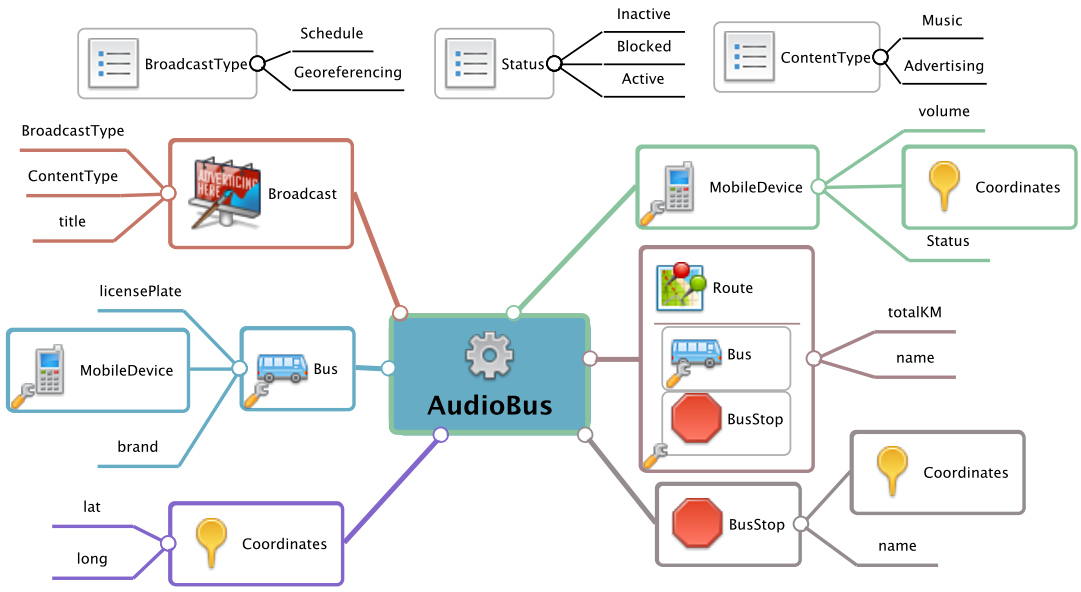


Figura 6.4: Exemplo de um modelo de domínio da linguagem *DomainMap*

Meta-Modelo

O meta-modelo do *DomainMap* é apresentado na Figura 6.5. O meta-modelo é composto por dois pacotes: *MindMapMetaModel* e o *ConceptualModelMetaModel*. O pacote *MindMapMetaModel* representa os elementos base de um mapa mental. Tem os elementos principais de um mapa mental: nó (*Node*) e arco (*Edge*). Um nó tem um nome, uma nota e uma notação visual. A notação visual serve para especificar a notação de domínio, sob a forma de ícone. A partir do elemento *Node* são especializados todos os outros nós, herdando assim a capacidade de modificar a sua notação visual. Num mapa mental existem dois tipos de nós, o nó raiz (*RootNode*) e um nó flutuante (*FloatNode*). O nó raiz representa

o centro do mapa mental, e possui uma ligação via arcos, direta ou indireta, com os restantes nós. Os nós flutuantes são os únicos nós que não possuem uma ligação direta ao nó raiz.

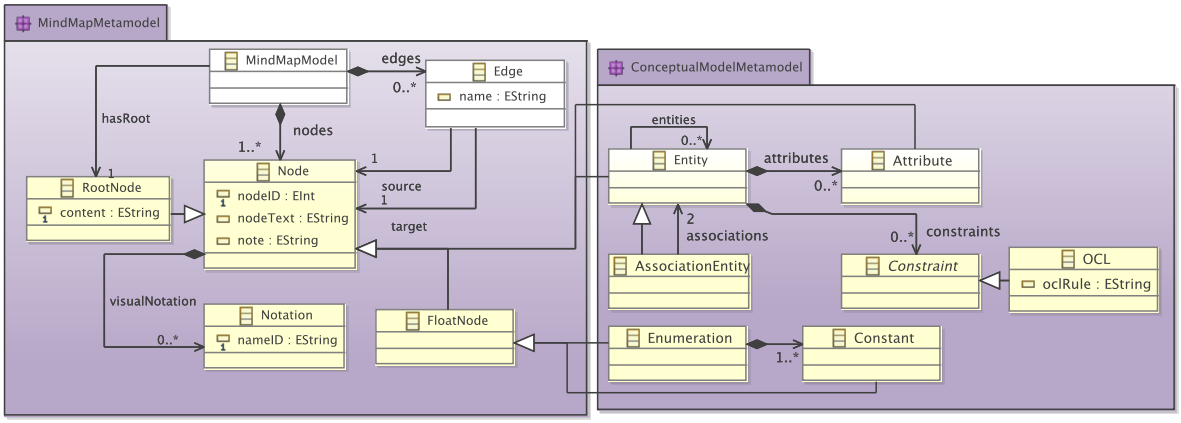


Figura 6.5: Meta Modelo do DomainMap

O segundo pacote, *ConceptualModelMetaModel* especifica os nós referentes a um modelo de domínio. A partir de nó (*Node*) são feitas especializações para elementos de um modelo de domínio: Entidade (*Entitity*), Entidade Associativa (*AssociationEntity*), Atributo (*Attribute*), Enumeração (*Enum*) e Constante (*Constant*). As entidades são compostas por um ou mais atributos ou outras entidades. Existem também entidades associativas que são especializações de entidades e têm de ter duas associações. Por fim, um enumerado é constituído por uma ou mais constantes.

Outra característica da ferramenta é a inserção de regras de negócio especificadas em OCL em nós entidades. Uma entidade tem várias regras OCL associadas. Cada regra OCL não tem qualquer validação sintática e de conteúdo, onde essa responsabilidade cabe ao utilizador que as insere. Esta falta de validação é um ponto de melhoria da ferramenta. No entanto, as regras OCL podem melhorar alguns aspectos no uso de transformações para código *Java*, explicado em detalhe na secção 6.2.1.

Restrições

Para garantir regras na construção do modelo de domínio que não conseguem ser garantidas através do meta-modelo, restrições foram escritas. As restrições podem recair em dois grupos: validações do mapa mental ou do modelo de domínio. A listagem 6.1 mostra a regra de validação que só pode existir um nó raiz. As restrições relativas ao mapa mental são:

1. Só pode existir um nó raiz;
2. Não podem existir nós sem nome;
3. Um nó só pode ter uma única palavra como nome;
4. Não podem existir nós que apontem para eles próprios;

5. Todos os nós tem de estar ligados, excepto se forem *Float Nodes*.

Listagem 6.1: Regra de Validação para só existir um único nó raiz

```
constraint OnlyOneRoot {
  check : self.select(f : Node | isCenter(f.type())).size() = 1
  message: 'There can only be one root type Node in a MindMap Model'
}
```

A listagem 6.2 mostra a regra que atributos só podem ser filhos de entidades. Relativamente ao modelo de domínio as restrições são:

1. O nó raiz só pode ter filhos do tipo Entidade;
2. Atributos só podem ser filhos de entidades;
3. Entidades só podem ser filhos do nó raiz ou de outras entidades;
4. Uma entidade associativa tem de ter duas entidades associadas;
5. Nós enumerados só podem ter filhos do tipo constante;
6. Uma entidade só pode ser definida uma única vez, i.e., caso existam duas entidades com o mesmo nome, apenas uma pode ter atributo definidos;
7. Não podem existir duas entidades iguais (i.e. com o mesmo nome) filhas do mesmo nó.

Listagem 6.2: Regra de Validação para que atributos só sejam filhos de entidades

```
constraint AttributesCanOnlyPointToEntity {
  check {
    var n = self.nodes.select(n : Node | isAttribute(n.type()) and
      self.edges.select(c: Edge | c.target = n and (isEntity(c.source.type()) or
      isAssociativeEntity(c.source.type()))).isEmpty());
    return n.isEmpty();
  }
  message : 'Atributes can only link to Entities: ' +
    transformSetToReadable(n+"")
}
```

Sintaxe concreta

Os elementos do modelo de domínio são apresentados na Tabela 6.1. A tabela apresenta os elementos gerais de qualquer modelo de domínio e os elementos especificados para o exemplo da Figura 6.4. Os elementos adotados foram escolhidos em colaboração com os *stakeholders* da *Mobciti*, executando o processo demonstrado na Figura 6.2.

Tabela 6.1: Notações visuais *DomainMap*

Conceito *DomainMap*	Notação visual por defeito	Conceito *Audiobus*	Notações visual do *Audiobus*
Nó Raiz		**Bus** (*Entidade*)	
Entidade		**Broadcast (***Entidade***)**	
Entidade Associativa		**BusStop** (*Entidade*)	
Enumerado		**MobileDevice** (*Entidade*)	
Atributo e Constante	Sem Notação	**Coordinates** (*Entidade*)	

Cenário de utilização

O funcionamento da ferramenta é baseado com interações através do rato e teclado. A ferramenta desenvolvida é baseada no ambiente *Eclipse* e tem uma barra lateral (assinalada a azul na Figura 6.6) com os elementos possíveis a adicionar na tela (assinalado a vermelho) onde se encontra o nó raiz.

Um modelo é iniciado com um nó raiz, que representa o sistema como um todo, exemplificado na Figura 6.6. As entidades do domínio podem ser criadas de múltiplas formas: arrastando da barra lateral para a tela, com menu *pop-up* ou aceleradores visuais da ferramenta. A barra lateral tem os elementos possíveis para adicionar ao editor, neste caso podemos adicionar, Entidades, Entidades Associativas, Atributos, Enumerados e Constantes.

Ao adicionar uma entidade, podemos adicionar uma descrição à mesma, regras OCL e alterar a sua notação visual. A Figura 6.7 mostra as opções possíveis. Quando uma entidade tem OCL ou descrições, aparecem notações específicas para notificar os leitores. Esse caso é exemplificado na Figura 6.8.

Transformações

De forma a transmitir as informações agrupadas no modelo de domínio para mais artefactos de desenvolvimento, foram desenvolvidas transformações para artefactos de código, concretamente código *Java*. A transformação para código *Java* cria classes para cada entidade definida, juntamente com os seus atributos a servirem de variáveis (com o tipo por defeito *String*). Um exemplo de transformação é apresentado na Figura 6.9. A classe *Bus* resulta da entidade *Bus* mostrada no exemplo, onde é criado um construtor vazio e para cada um dos seus filhos (atributo ou entidade) são criadas variáveis na sua classe,

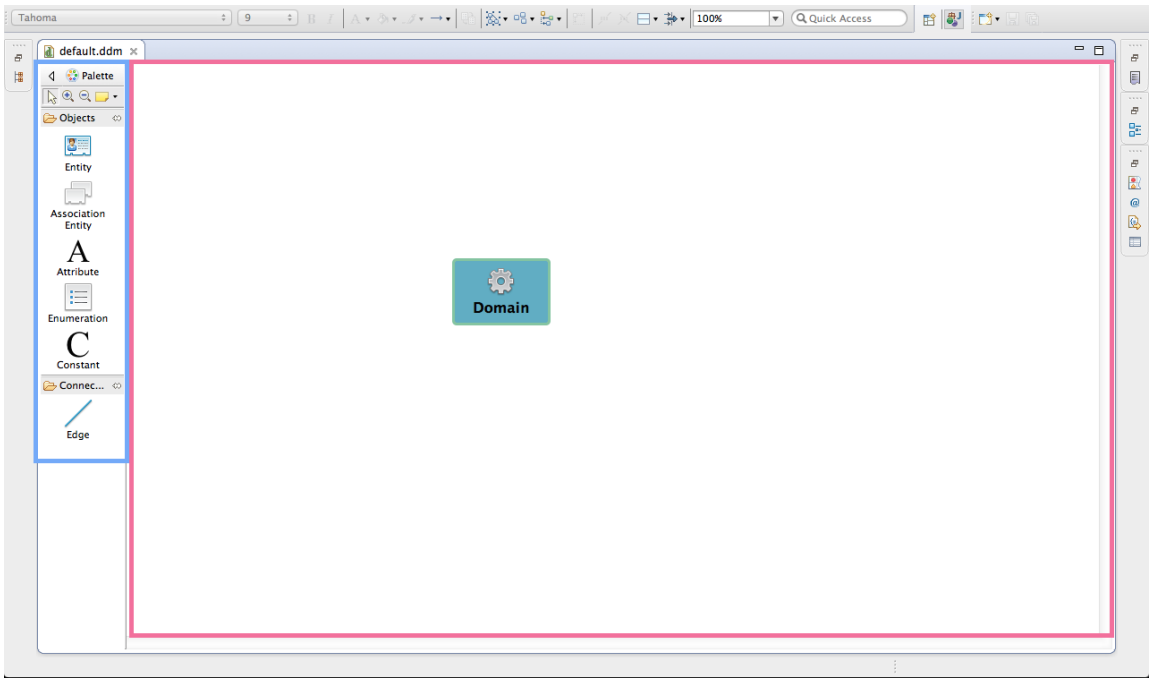


Figura 6.6: *DomainMap* - Modelo Inicial

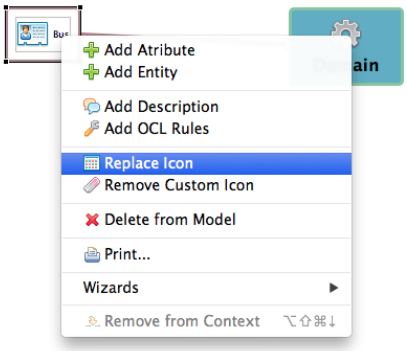


Figura 6.7: Opções de interacções com um nó do tipo Entidade

Figura 6.8: Modelo com entidade *Bus* com regras OCL e Descrição

cada um com um método *getter* e *setter*. O código Java gerado é mostrado na Listagem 6.3. O código da transformação está presente no anexo C.4.

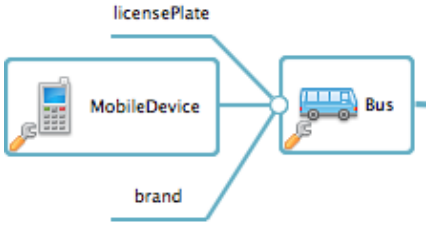


Figura 6.9: Exemplo Modelo Conceptual

Como existe a possibilidade de introduzir regras OCL a cada entidade, essa restrição OCL pode servir para poder inferir o seu tipo na transformação. A inferência é feita procurando por operadores relacionais[2] e analisando a variável que precede e o valor da restrição que procede. Caso a restrição mencione inteiros, a variável será *Integer*; caso seja mencionado um decimal o resultado será *Double*; caso seja um valor booleano (*True* ou *False*) o tipo será *Boolean*. Esta funcionalidade é ainda embrionária e carece de muito desenvolvimento para limitar os erros existentes. Foi desenvolvida para mostrar os benefícios que a linguagem pode introduzir.

Listagem 6.3: Transformação gerada para o exemplo da Figura 6.9

```
/**
* Bus.java  1.0 Sun Mar 23 23:07:42 WET 2014
* Created by: MMClass
* Property of FCT-UNL
*/

/**
* Bus
* @version 1.0 Sun Mar 23 23:07:42 WET 2014
* @author: MMClass
*/
public class Bus {
  private String brand;
  private String licensePlate;
  private MobileDevice mobileDevice;

  public Bus() {
    //TODO
  }
  public void setBrand(String value) {
    brand = value;
  }
  public String getBrand() {
    return brand;
```

[2] =, >, <, ≥, ≤, <>

```
  }
  public void setLicensePlate(String value) {
    licensePlate = value;
  }
  public String getLicensePlate() {
    return licensePlate;
  }
  public void setMobileDevice(MobileDevice value) {
    mobileDevice = value;
  }
  public MobileDevice getMobileDevice() {
    return mobileDevice;
  }
}
```

6.2.2 BehaviorMap - Linguagem para Modelos Comportamentais BDD

A segunda e principal linguagem da dissertação, denominada *BehaviorMap*, serve para especificar cenários comportamentais BDD graficamente utilizando mapas mentais. Os cenários são representados com um estado inicial, uma mudança nesse mesmo estado e um estado final. A linguagem consegue importar o modelo de domínio criado com a linguagem, *DomainMap*, apresentada na secção 6.2.1. Através dessa incorporação, as entidades, as suas notações visuais e atributos são incorporados na construção do cenário. Com isto, as notações criadas com a colaboração dos *stakeholders* são utilizadas na definição de cenários.

Um exemplo produzido com a ferramenta é apresentado na Figura 6.10. Esta representa o cenário *BroadcastSpecificArea* relacionado com a empresa *Mobciti*. O exemplo representa o comportamento que um autocarro (*Bus*) deve ter ao passar pelas coordenadas 34º53.720, -8º3.071. Quando o autocarro passar por esses coordenadas, deve alterar a música que está a transmitir para um anúncio do *MacDonalds*, mudar o tipo de anúncio para *GeoReference* e o tipo de conteúdo para *Advertising*. Além disso, deve aumentar o volume do *MobileDevice* para *0.9*.

Estrutura da linguagem

O objetivo principal da linguagem é conseguir especificar o estado de um conjunto de entidades antes e depois de uma (ou mais) ações ocorrerem. A Figura 6.11 mostra os conceitos em que o *BehaviorMap* se baseou. Existe um estado inicial (*EI*), uma ação (*F*) e um estado final (*EF*), resultado de aplicar a ação *F* em *EI*. Tanto o estado inicial e final têm acesso a um e único *Agente* no cenário. A função *F* pode ser apenas executada por entidade (ou agente) pertencentes ao *EI* e recebe como parâmetros valores básicos *String*, *Booleano*, *Integer* e *Double* ou uma entidade pertencente ao *EI*. No *EI* são feitas as assunções iniciais que se esperam ter antes de executar a função *F* e no *EF* são feitas asserções finais para garantir os resultados esperados.

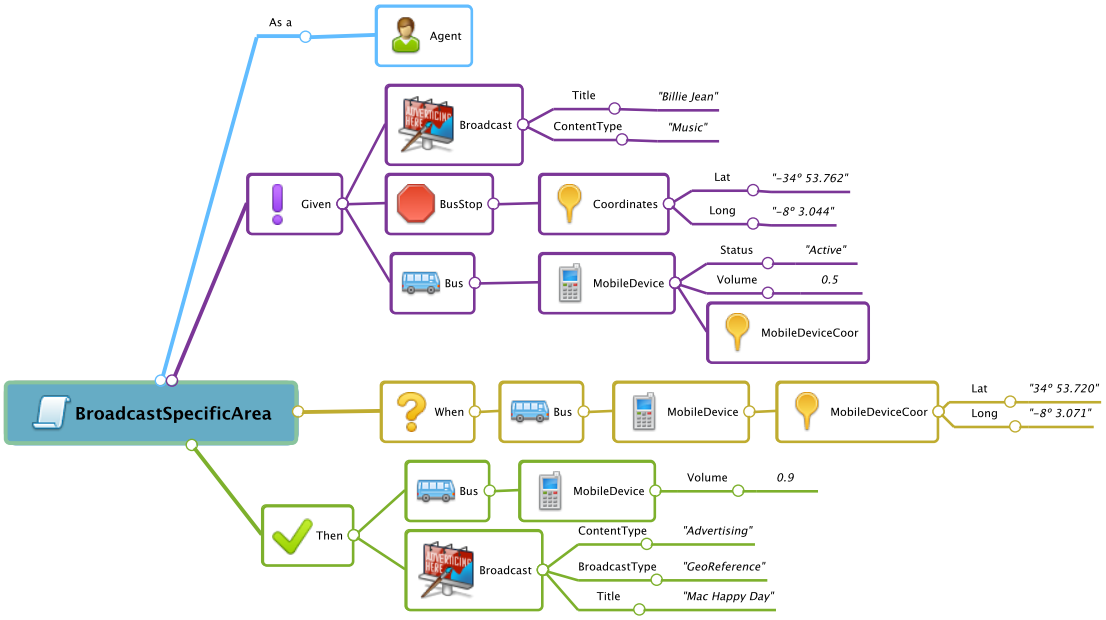


Figura 6.10: Exemplo de um modelo comportamentel da linguagem *BehaviorMap*

No *EI* apenas se podem especificar entidades, os seus atributos e respetivo valor. Relativamente ao *EF*, para além da especificação de entidades, existem considerações a ter em conta. Existem casos de comportamentos que a execução de uma ação de uma certa entidade pode despoletar a execução de outras ações por parte de outras entidades. Caso queiramos representar esses casos é necessário que no *EF* as entidades mencionadas consigam especificar a execução de uma ação e o seu resultado esperado, logo o *BehaviorMap* permite o mesmo. Uma ação que é especificada no *EF*, tem necessariamente de especificar um resultado esperado, de forma que seja verificável num teste. No caso da ação *F*, ação especificada no ramo *When*, os seus elementos descendentes são considerados parâmetros de entrada, para poder aumentar a expressividade da linguagem.

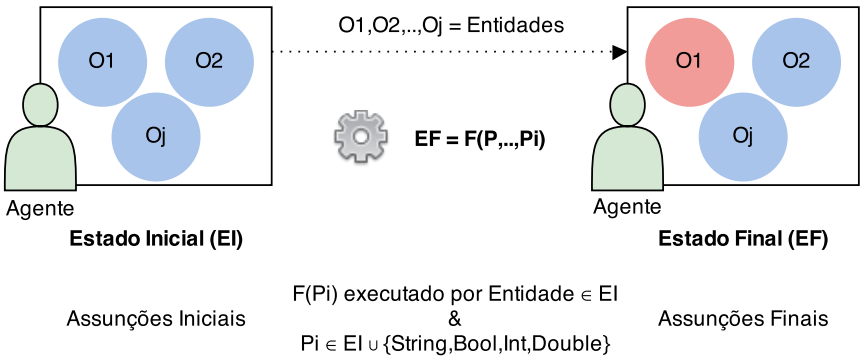


Figura 6.11: Estrutura do *BehaviorMap*

Este tipo de abordagem tem uma limitação em relação à remoção de entidades. No

BDD existe uma regra que afirma que tudo o que não seja afirmado no passo *Then* permanece igual ao especificado no passo *Given*. Logo caso uma entidade não seja mencionada no estado final, significa que não foi afetada pela execução da ação. Uma forma de contornar este problema é associar um atributo *estado* a uma entidade e mudar o seu valor para *removido* para simular a remoção.

A verificação de cenários comportamentais foi baseado na Lógica de *Hoare*. Na Lógica de *Hoare* temos uma tripla com a forma '{*P*} *C* {*Q*}', em que *P* e *Q* são asserções e *C* é um comando. Um triplo de *Hoare* lê-se da seguinte forma: "Sempre que as asserções *P* forem verdadeiras antes da execução de *C*, então, após a execução de *C*, as asserções *Q* têm de ser verdadeiras".

Os cenários do *BehaviorMap* podem ser vistos de forma semelhante, como ilustra a Figura 6.12. As condições esperadas no passo *Given* são semelhantes às asserções *P*, as ações no passo *When* que correspondem ao comando *C* e o resultado esperado no passo *Then* que correspondem às asserções *Q*. A tripla forma um cenário comportamental.

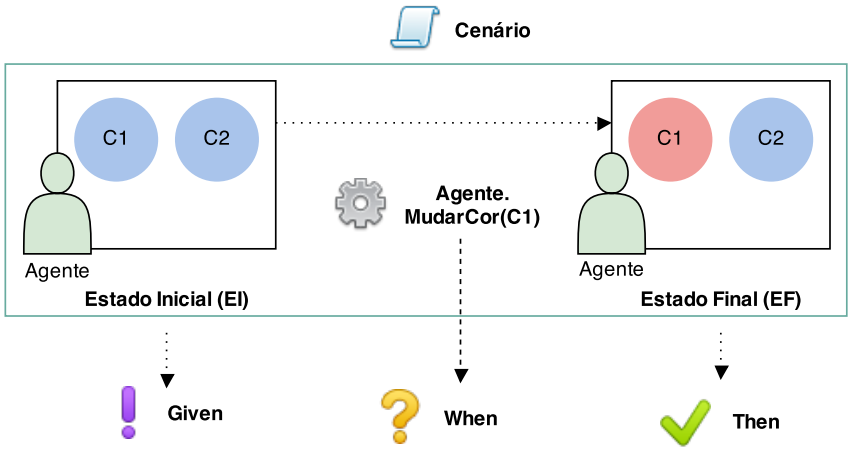


Figura 6.12: Correspondência com o BDD

Meta-Modelo

O meta-modelo da ferramenta está apresentado na Figura 6.13. O meta-modelo é composto por três pacotes principais: *MindMap*, *MindMapBDD* e *ConceptualModelInegration*. Tal como o meta-modelo do modelo de domínio, os elementos essenciais do mapa mental estão presentes no pacote *MindMap*.

A partir do Nó (*Node*) são especializados, no pacote *MindMapBDD*, os nós que constituem os elementos necessários para representar o comportamento: *Given* (*Given*), *When* (*When*), *Then* (*Then*), Entidades (*Entity*), Atributo (*Attribute*), Valor (*Value*), Ação (*Action*). Esses elementos compõem o pacote de elementos de um cenário BDD. Os elementos constantes de um cenário são o título do cenário (*RootNode*), o Agente (*Agent*), os três passos (*Given*, *When* e *Then*). Cada passo pode ser constituído por um Agente ou Entidades. Cada Entidade é composta por Atributos, outras Entidades ou Ações. Cada Atributo tem

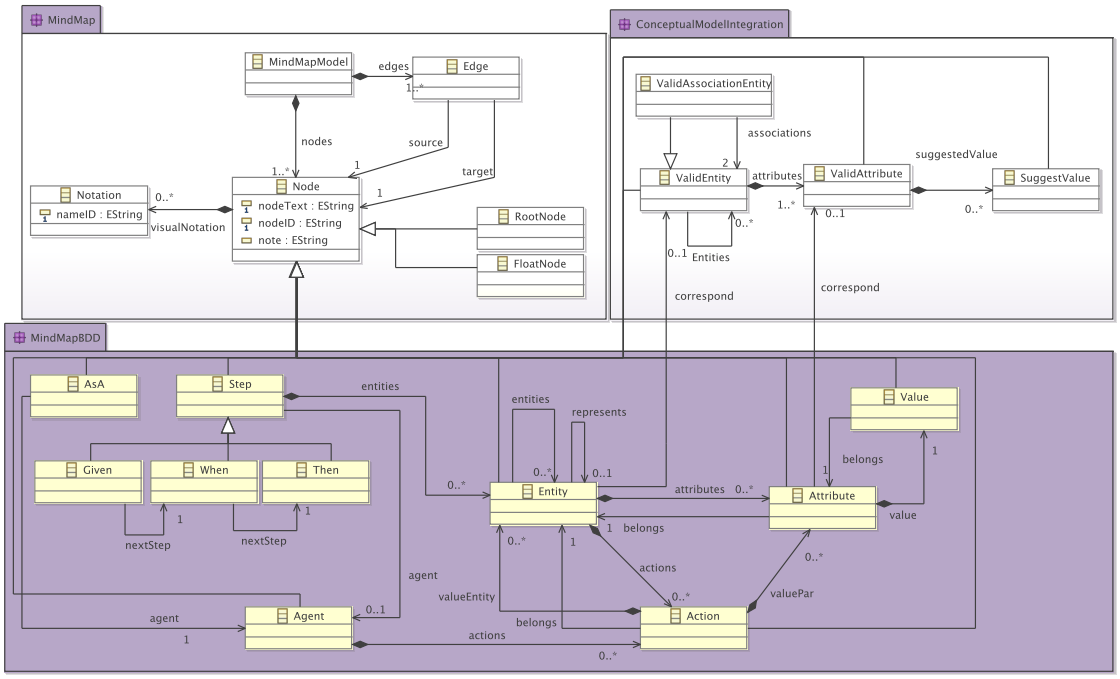


Figura 6.13: Meta Modelo da linguagem BehaviorMap

um e um só Valor. As Ações conseguem receber como parâmetro atributos ou entidades. Um Agente apenas realiza Ações. As Entidades são definidas no passo *Given*, sendo o seu nome (*nodeText*), o seu identificador único no cenário. Uma Entidade pode ser referenciada em passos distintos no cenário. De forma a saber se uma Entidade representa outra, existe a ligação *represents*.

O último pacote possui os elementos para possibilitar a integração do modelo de domínio com o cenário. A integração serve como uma forma de propagar a informação capturada e validada no modelo de domínio para servir como base de construção dos cenários comportamentais. O pacote é constituído por *validEntity* que é composta por *validAtributes*. Cada *validAtribute* tem vários *suggestValues*. Existe também o elemento *validAssociationEntity*, especialização de *validEntity* para representar entidades associativas. Esses elementos são carregados com as respetivas entidades e os seus atributos quando um modelo de domínio é importado. O elemento *suggestValue* serve para sugerir valores interessantes para criar testes de aceitação com base em regras OCL especificadas. No entanto, essa funcionalidade não foi implementada devido a restrições temporais. A ligação *correspond* serve para fazer a ligação entre uma *Entity* e um *validEntity* e um *Atribute* e um *validAtribute*.

Restrições

De forma a garantir regras na construção dos cenários comportamentais que não conseguem ser expressas através da meta-modelação e simultaneamente auxiliar a especificação de um modelo comportamental, restrições foram criadas. Estas partilham as regras

comuns ao de um mapa mental, apresentadas para o *DomainMap*: tem de existir um e um só nó central, todos os nós tem de estar ligados por um arco, todos os nós têm de ter um único nome e não podem existir espaços num nome (com a exceção do elemento *As a*). A listagem 6.4 apresenta a regra que impede a criação de nós, além de entidades, no passo *Given*. As restrições introduzidas no *BehaviorMap* são:

1. No passo *Given* tem de existir pelo menos uma entidade definida;
2. No passo *Given* não podem existir ações;
3. No passo *When* tem de existir pelo menos uma ação ou mudança de atributo definida;
4. No passo *Then* tem de existir pelo menos uma entidade definida ou uma ação;
5. Todas as entidades no passo *When* tem de ter um representante no passo *Given*;
6. Não podem existir duas entidades com o mesmo nome filhas do mesmo passo;
7. No passo *When*, os parâmetros do tipo entidade de uma ação têm de ter valores definidos:
   (a) Se a entidade não tiver representante tem de ter atributos com valores definidos;
   (b) Se a entidade for representante, não pode ter atributos definidos;
8. Os agentes só podem cometer ações;
9. Só existe um agente, com um único nome.

Listagem 6.4: Regra de Validação para que só existam entidades filhas no passo *Given*

```
constraint GivenMustPointToEntities {
  check {
    var n = self.nodes.select(n : Node | isGiven(n.type()) and
      self.edges.select(c: Edge | c.source = n and
      not (isEntity(c.target.type()))).size() > 0);
    return n.isEmpty();
  }
  message : 'Step Given can only link to Entities: ' +
       transformSetToReadable(n+"")
}
```

Também foram adicionados avisos para ajudar na construção dos cenários:

1. É conveniente que uma entidade no passo *Then* tenha algum atributo definido;
2. É conveniente que cada ação especificada no passo *Then* tenha uma ou mais expectativas;
3. Ao existir uma entidade associativa, avisar caso os associados não estejam criados;
4. Cada entidade tem de ter um tipo definido.

Sintaxe concreta

Os elementos do *BehaviorMap* são apresentados na Tabela 6.2. As entidades são os únicos elementos que podem sofrer alterações na sua notação visual. Essa alteração nasce da importação do modelo de domínio proveniente do *DomainMap*. Com a importação do modelo de domínio as entidades passam a ser de vários tipos, de acordo com as diferentes entidades criadas num modelo de domínio.

Tabela 6.2: Notações visuais *BehaviorMap*

Conceito cenário BDD	Notação visual por defeito
Given	
When	
Then	
Nó Raiz	
Entidade	
Agente	
Atributo	Sem Notação
Valor	*Escritos em itálico*

Cenário de utilização

O *BehaviorMap* funciona de forma semelhante ao *DomainMap*, através de interações com o rato e teclado. A ferramenta inicia com os elementos constantes a qualquer cenário BDD: um nó com o título, um agente e os três passos (*Given*, *When* e *Then*). A partir de cada passo, são criadas entidades através de várias formas: da barra lateral, com menu *pop-up* e aceleradores visuais da ferramenta. Apenas o menu *pop-up* não é criado por defeito pelo EuGENia e teve de ser criado através de uma extensão. A Figura 6.14 mostra um diagrama inicial, e a Figura 6.15 um *pop-up* para adicionar um elemento e o seu resultado.

A Figura 6.16 mostra o ambiente da ferramenta ao importar um modelo conceptual, vindo do *DomainMap*. A barra lateral altera-se com a adição das entidades e os seus respetivos ícones, como mostra a Figura 6.16. A Figura 6.16 mostra também uma entidade denominada *Bus* à qual se vai se dar o tipo *Bus*, tipo esse que foi definido no modelo conceptual apresentado na Figura 6.4.

Ao alterar o tipo da entidade '*Bus*' para *Bus* alteramos mais que a sua notação visual. Ao adicionar um atributo a uma dessas entidades, existe a possibilidade de ter sugestões dos nomes de atributos a inserir. Ao pressionar na tecla '_' um menu de sugestão surge

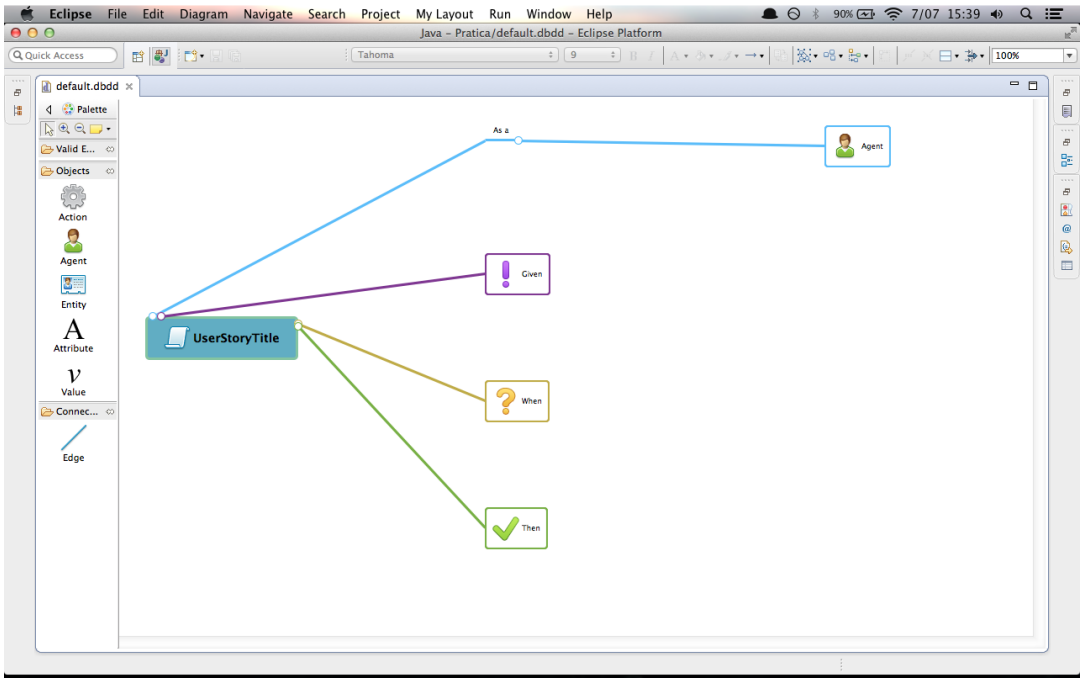


Figura 6.14: Diagrama inicial apresentado na ferramenta *BehaviorMap*

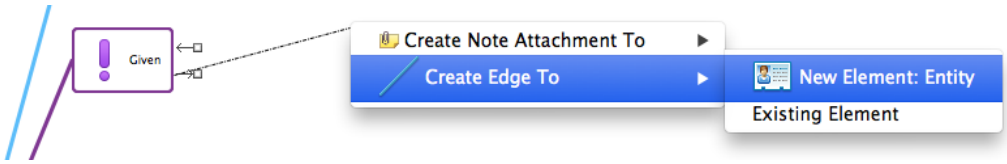


Figura 6.15: Adicionar uma nova entidade ao ramo Given no *BehaviorMap*

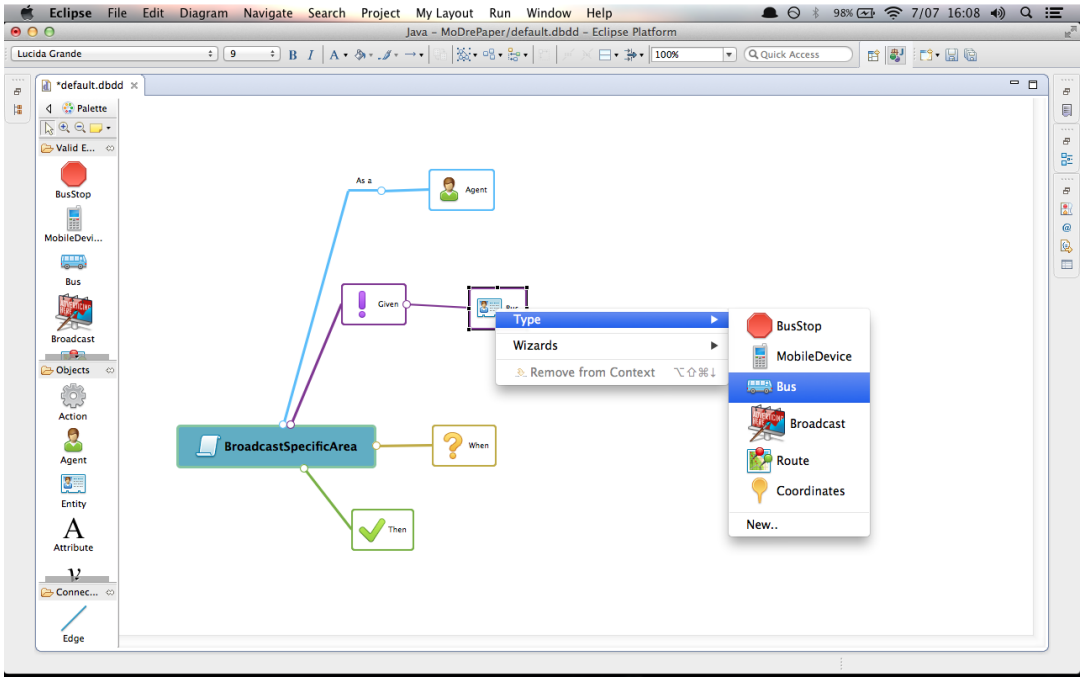


Figura 6.16: Mudar tipo de um entidade para *Bus*

com os atributos que foram definidos no modelo de domínio sobre essa entidade. Esta funcionalidade também está disponível para sugerir nomes de entidades quando uma entidade é composta por outras. A Figura 6.17 ilustra a funcionalidade.

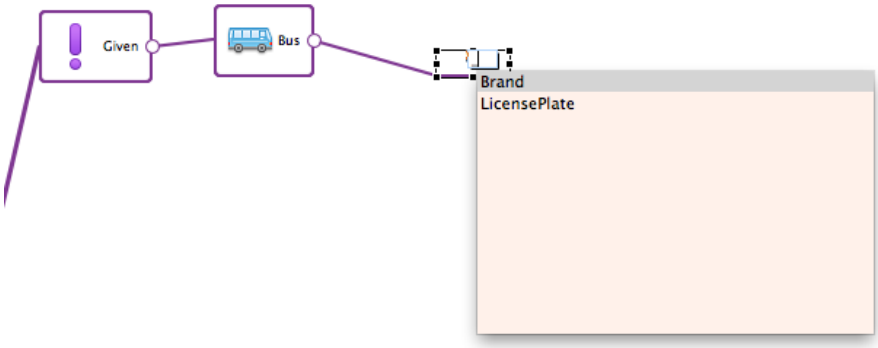


Figura 6.17: Sugestão de nomes de atributos para a entidade *Bus*

Ao adicionar um valor a cada atributo a ferramenta consegue fazer distinção entre valor do tipo *String* e números. Se um carácter corresponder a uma letra ou um espaço são inseridas aspas no início e no fim do valor, caso contrário é assumido como número. A Figura 6.18 mostra um exemplo de diversos valores.

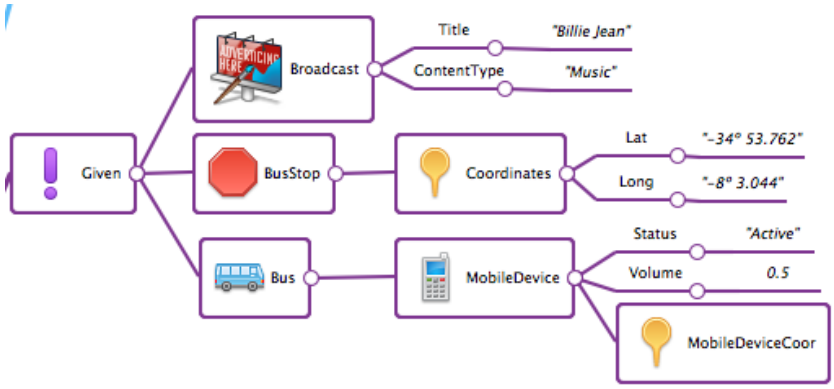


Figura 6.18: Exemplo da sintaxe concreta de atributos

O auxílio de sugestões também está disponível na definição de entidades. Como as entidades são definidas no passo *Given*, as suas referências são utilizadas nos passos seguintes. Ao selecionar uma entidade criada no passo *When* e *Then* e pressionar em '_' surge uma lista com as entidades definidas no passo *Given*. Ao alterar o nome da entidade para uma entidade definida no passo *Given* o seu ícone e tipo são alterados, tal como a Figura 6.19 exemplifica.

Para não restringir a forma de utilização da ferramenta, existe a possibilidade de utilizar a ferramenta sem um modelo de domínio como base. Essa decisão tem algumas implicações. Não existe a capacidade de ter a funcionalidade de sugestão de nomes, nem de adicionar ícones personalizados.

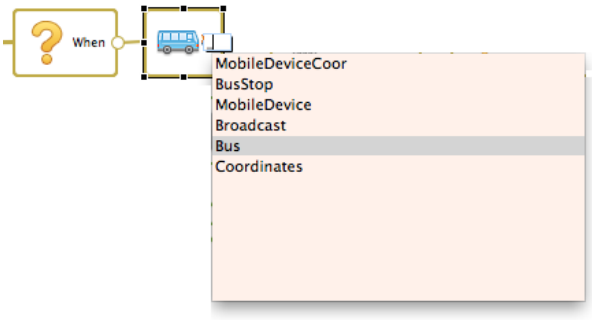


Figura 6.19: Auto-complete para entidades

Transformações

Para garantir que o comportamento especificado nos cenários *BehaviorMap* consegue ser validado no ato de produção de código, foi criado um método de transformação de cenário *BehaviorMap* para teste de aceitação. O alvo da transformação é a plataforma de testes *JUnit*. A estrutura do teste é apresentado na Listagem 6.5. Cada teste tem uma função *Given*, *When* e *Then*, para representar cada passo do teste. A cada uma delas vão ser mapeados os elementos criados no cenário. O código ETL utilizado na transformação está apresentado no anexo C.3.

Listagem 6.5: Exemplo de classe de teste na plataforma *JBehave*

```
public class ScenarioTitle {
  @Rule
  public ErrorCollector collector = new ErrorCollector();
  @Before
  public void given() {
    /*ASSERTIONS*/
    /*--*/
  }
  @Test
  public void when() {
    then();
  }
  public void then() { }
```

A Listagem 6.6 exemplifica o código transformado a partir do exemplo mostrada na Figura 6.20. As entidades especificadas no passo *Given* são criadas e inicializadas na respetiva função no ficheiro *JUnit*. Para cada inicialização é chamada uma função *setter* para o atributo em questão. Além disso, é criada uma asserção para cada inicialização de forma a garantir que o estado inicial está de acordo com as especificações.

Listagem 6.6: Exemplo de classe de teste na plataforma *JBehave*

```
@Before
public void given() {
  bus = new Bus();
  mobileDevice = new MobileDevice();
```

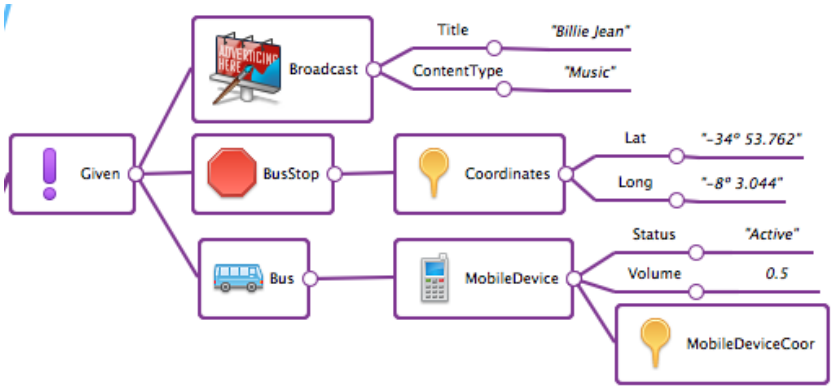


Figura 6.20: Passo *Given* de um cenário comportamental

```
  mobileDeviceCoor = new Coordinates();
  broadcast = new Broadcast();
  busStop = new BusStop();
  coordinates = new Coordinates();

  mobileDevice.setCoordinatesMD(mobileDeviceCoor);
  mobileDevice.setVolume(0.5);
  mobileDevice.setStatus("Active");

  bus.setMobileDevice(mobileDevice);
  ...
  /*ASSERTIONS*/
  collector.checkThat(mobileDeviceCoor,
    CoreMatchers.equalTo(mobileDevice.getMobileDeviceCoor()));
  collector.checkThat(0.5,
    CoreMatchers.equalTo(mobileDevice.getVolume()));
  collector.checkThat("Active",
    CoreMatchers.equalTo(mobileDevice.getStatus()));
  collector.checkThat(mobileDevice,CoreMatchers.equalTo(
    bus.getMobileDevice()));
  ...
  /*--*/
}
```

No passo *When* são criadas funções para representar as mudanças que acontecem a qualquer entidade mencionada nesse mesmo ramo. A Figura 6.21 mostra que apenas a entidade *MobileDeviceCoor* é alterada e a Listagem 6.7 apresenta o código do teste gerado.

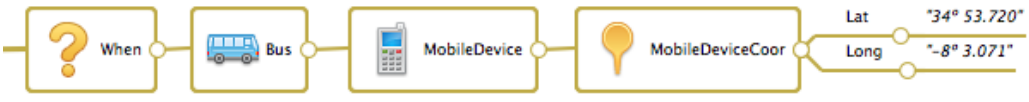


Figura 6.21: Passo *When* de um cenário comportamental

Listagem 6.7: Exemplo de classe de teste na plataforma *JBehave*

```
@Test
```

```
public void when() {
  /*The next functions need to be implemented in order to
     have the expected outcome*/
  whenBus();
  whenMobileDevice();
  whenMobileDeviceCoor();
  /* -- */
  /* The next variables were created here because they
    are probably related to the previous actions*/
  /* -- */
  then();
}
public void whenBus() { }
public void whenMobileDevice() { }
public void whenMobileDeviceCoor() {
  mobileDeviceCoor.setLat("34   53.720");
  mobileDeviceCoor.setLong("-8   3.071");
}
```

No caso do passo *Then* são mapeadas as entidades especificadas e criadas asserções para verificar os atributos especificados. Contudo, qualquer atributo não especificado no passo *Then* tem de manter o seu valor original, logo é inserida uma asserção para cada atributo que permaneceu inalterado. Caso uma ação seja especificada no *Then*, o mapeamento para o teste é feito com uma asserção onde o seu retorno é esperado sob a forma de uma asserção. A Figura 6.22 mostra o cenário utilizado para o código de teste gerado mostrado na Listagem 6.8.

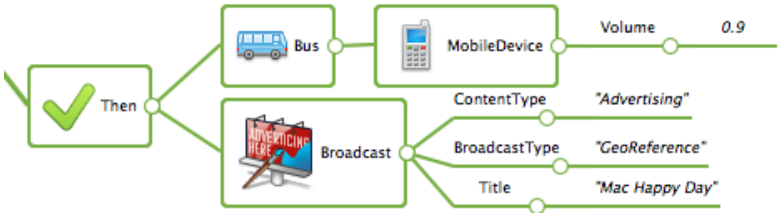


Figura 6.22: Passo *Then* de um cenário comportamental

Listagem 6.8: Exemplo de classe de teste na plataforma *JBehave*

```
public void then() {
  collector.checkThat(0.9,CoreMatchers.equalTo(mobileDevice.getVolume()));
  collector.checkThat(mobileDevice,CoreMatchers.equalTo(
    bus.getMobileDevice()));

  /*Check if all remains the same*/
  /* -- */

  collector.checkThat("Mac Happy Day",CoreMatchers.equalTo(
    broadcast.getTitle()));
  collector.checkThat("Advertising",CoreMatchers.equalTo(
    broadcast.getContentType()));
```

```
13    collector.checkThat("GeoReference",CoreMatchers.equalTo(
        broadcast.getBroadcastType()));
15
      /*Check if all remains the same*/
17    /* -- */

19    /*Check if all remains the same*/
      collector.checkThat(mobileDeviceCoor,CoreMatchers.equalTo(
21      mobileDevice.getMobileDeviceCoor()));
      collector.checkThat("Active",CoreMatchers.equalTo(mobileDevice.getStatus()));
23    /* -- */
  }
```

Casos especificos

O cenário apresentado na Figura 6.23 especifica um comportamento sobre uma ação do agente *Coast Guard* ao visualizar um mapa, dado dois navios com diferentes velocidades.

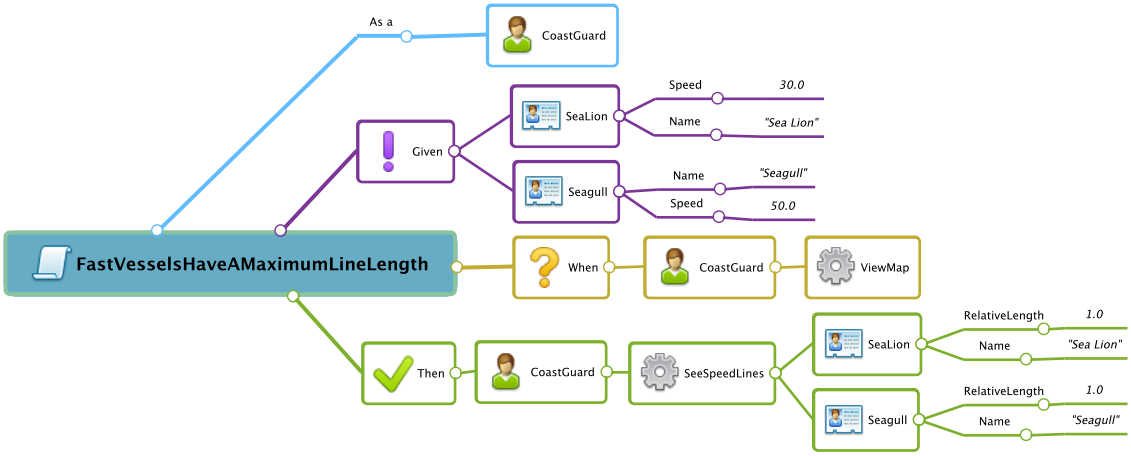


Figura 6.23: Cenário *BehaviorMap - Fast vessels have a maxinum line lenght*

Ao transformar este cenário num teste *JUnit*, é necessário ter atenção ao mapeamento das ações no passo *Then*. Neste exemplo a ação *SeeSpeedLines* retorna duas entidades. O código de transformação gerado é mostrado na Listagem 6.9. Como se pode verificar é necessário adaptar o código para que os elementos sejam retornados numa só estrutura de dados. No entanto, a ferramenta escreve um comentário como o indicado na Listagem 6.9 para notificar o utilizador da necessidade de modificação. Esta melhoria não foi realizada devido a escassez de tempo.

Listagem 6.9: Exemplo de classe de teste na plataforma *JBehave*

```
public void coastGuardMustSeeSpeedLines() {
2   /*REFINEMENT IS NEEDED BECAUSE ACTION WITH MULTIPLE RETURNS*/
    Vessel seaLion = coastGuard.seeSpeedLines();
4   collector.checkThat("Sea Lion",CoreMatchers.equalTo(seaLion.getName()));
    collector.checkThat(1.0,CoreMatchers.equalTo(seaLion.getRelativeLenght()));
```

```
6   /*Check if all remains the same*/
    collector.checkThat(30.0,CoreMatchers.equalTo(seaLion.getSpeed()));
8   /* -- */

10  Vessel seagull = coastGuard.seeSpeedLines();
    collector.checkThat("Seagull",CoreMatchers.equalTo(seagull.getName()));
12  collector.checkThat(1.0,CoreMatchers.equalTo(seagull.getRelativeLenght()));
    /*Check if all remains the same*/
14  collector.checkThat(50.0,CoreMatchers.equalTo(seagull.getSpeed()));
    /* -- */
16  }
```

Outro possível problema é a ordenação na escrita das ações nos testes gerados. As ações na transformação são escritas pela ordem que são capturadas no *script* de transformação, ou seja não existe nenhuma ordem especifica de chamada de execução. Isto pode ser um problema caso uma ação afete outra. Este problema foi notado, mas não foi resolvido devido ao tempo disponível.

6.3 Resumo

Neste capítulo foram mostradas as duas linguagens desenvolvidas nesta dissertação e as suas implicações no processo de desenvolvimento de *software* do BDD. Foi proposto uma alteração ao processo BDD para incorporar as linguagens desenvolvidas. As linguagens propostas pelas dissertação, o *BehaviorMap* e o *DomainMap* foram apresentadas, explicando o seu propósito e o seu objetivo no processo de desenvolvimento.

7

Avaliação

Este capítulo tem o propósito de expor ao leitor as avaliações realizadas para as linguagens propostas e desenvolvidas na dissertação. Foram realizadas duas avaliações para a linguagem *BehaviorMap* em termos de aplicabilidade e compreensão. A avaliação de aplicabilidade teve o objetivo de verificar se o *BehaviorMap* conseguia representar diversos cenários textuais de diversas fontes de trabalho. Já a avaliação de compreensão pretendia testar a hipótese se os cenários *BehaviorMap* são mais facilmente compreendidos em relação aos cenários textuais.

7.1 Aplicabilidade

Esta avaliação teve como princípio descobrir se a linguagem *BehaviorMap* consegue representar diversos cenários BDD textuais. De forma a verificar esta hipótese foi necessário recolher múltiplos cenários de diversas fontes [Lop11; NEČ30; Tav+10; Die+12; CIG13; Che+10; Nor13; Nor14]. Os cenários foram recolhidos de fontes utilizadas na revisão da literatura sobre a adopção do BDD, apresentado na secção 5. O processo de recolha foi feito da seguinte forma: o cenário era encontrado, copiado e inserido num repositório. No total foram encontrados 63 cenários. Cada cenário textual foi traduzido para *BehaviorMap* segundo um conjunto de regras apresentado na secção 7.1.1. Caso um cenário não conseguisse ser traduzido a razão era apontada para futura análise.

7.1.1 Tradução de cenários textuais para BehaviorMap

Para verificar se um cenário textual conseguia ser mapeado para o *BehaviorMap* foram feitas traduções. As traduções consistiam em capturar o comportamento expresso num

cenário textual para um cenário gráfico. As regras de tradução consistiam em encontrar substantivos, verbos e adjetivos para representar os conceitos presentes no *BehaviorMap*. Para o passo *Given* as regras foram:

1. Encontrar um substantivo para representar uma entidade;
2. Para cada substantivo encontrado, verificar se tem adjetivos para se tornarem atributos;
3. Para cada adjetivo encontrado, procurar o respectivo valor.

Relativamente ao passo *When*:

1. Encontrar um substantivo para representar uma entidade defina no passo *Given*;
2. Encontrar um verbo que identifique a ação que vai ser representar;
3. Verificar se essa ação tem algum valor necessário para ser transformado em parametro.

Respetivamente ao passo *Then*:

1. Encontrar um substantivo para representar uma entidade;
2. Para cada substantivo encontrado, verificar se tem adjetivos para se tornarem atributos;
3. Para cada adjetivo encontrado, procurar o respectivo valor;
4. Verificar a existência de algum verbo associado a um substantivo. Caso exista, alguma entidade comete alguma ação e espera um resultado.

Tomando como exemplo o cenário textual apresentado na Listagem 7.1. No passo *Given* consegue-se identificar o substantivo *Vessel* chamado *Seal* com adjetivo *position* com o valor '52.01N, 3.99E'. No passo *When* existe referência ao *I*. Cada vez que um *I* é utilizado, refere-se ao agente mencionado no cabeçalho na parte do *As a*. Neste caso, o agente é o *Coast Guard*. O verbo que indica a ação é *View*, mas na tradução a acção resultante tornou-se *ViewMap* e recebe uma Área com duas posições os dois parâmetros '52.10N, 3.90E' e '51.90N, 4.10E'. No passo *Then* existe novamente a referencia ao *I*, que resulta no agente a executar a ação *See* que retorna um *Vessel* na posição '52.01N, 3.99E'. A Figura 7.1 mostra o resultado da tradução.

Listagem 7.1: Cenário textual - *Map view*

```
Feature: Map View
In order to asses the situation in my area
As a coast guard
I want to see the location of each vessel marked on a map

Scenario: show vessel inside map area
  Given vessel "Seal" at position "52.01N, 3.99E"
  When I view the map area between "52.10N, 3.90E" and "51.90N, 4.10E"
  Then I should see vessel "Seal" at position "52.01N, 3.99E"
```

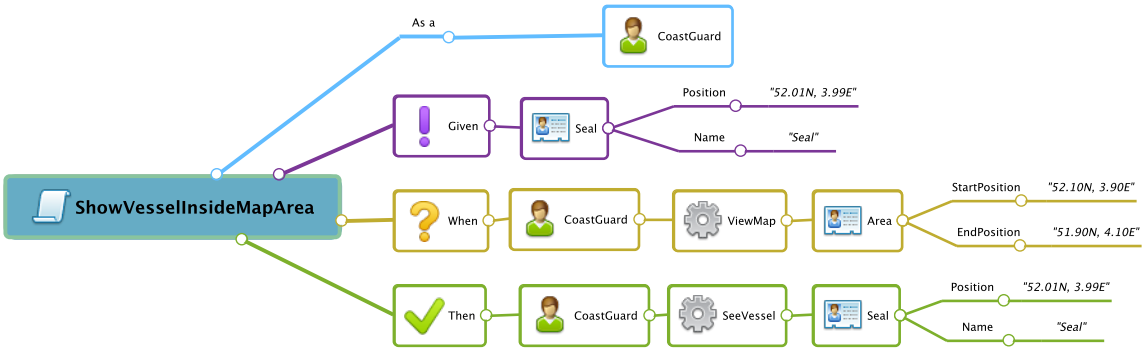


Figura 7.1: Cenário *Map view* escrito em *BehaviorMap*

O exemplo 7.2 mostra um exemplo de um cenário mais complicado de traduzir, devido a não ter toda a informação disponível e as decisões que foram tomadas. No cenário existe referência ao agente (utilizando o *I*), no entanto o agente não é especificado. Na tradução, foi criada uma entidade para representar a entidade *I* respeitando as regras do *BehaviorMap* e não modificando o comportamento. Em casos semelhantes de falta de informação para especificar o cenário, foram tomadas decisões semelhantes.

Listagem 7.2: Cenário textual - *Withdraw fixed amount of* 50

```
Scenario: Withdraw fixed amount of $50
  Given I have $500 in my account
  When I choose to withdraw the fixed amount of $50
  Then I should receive $50 cash
  And the balance of my account should be $450
```

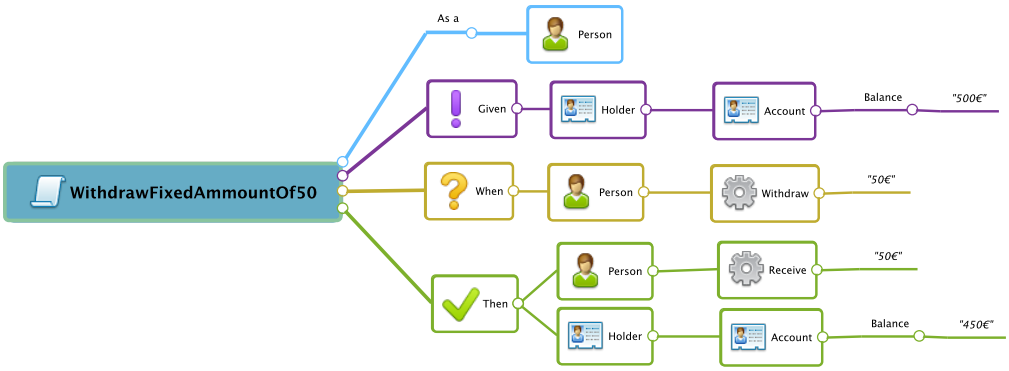


Figura 7.2: Cenário *Holder withdraws cash* escrito em *BehaviorMap*

Dos 63 cenários, 53 conseguiram ser mapeados para o *BehaviorMap*. As Listagens 7.3 e 7.4 mostram um cenário que não foi traduzido, devido à incapacidade de especificar condições temporais e por não ter o passo *When* e outro devido a ter mais que uma sequência de passos *When* e *Then*, respetivamente. As razões pelas quais 10 cenários não conseguiram ser mapeados para o *BehaviorMap* foram:

1. Cenários mal especificados (e.g. sem valores especificos de atributos ou acções), sem passo *When*, cenários a especificar aspectos de *interfaces*, múltiplos passos *When Then*;
2. Cenários com codições temporais (e.g. dado que três segundos passaram).

Listagem 7.3: Cenário textual não traduzido - *Reset button*

```
Feature: Reset button

Scenario: Reset while running
  Given 3 seconds have elapsed
  Then the clock should read "00:03"
```

Listagem 7.4: Cenário textual não traduzido

```
Given a 5 by 5 game
When I toggle the cell at (2, 3)
Then the grid should look like
.....
.....
.....
..X..
.....
When I toggle the cell at (2, 4)
Then the grid should look like
.....
.....
.....
..X..
..X..
When I toggle the cell at (2, 3)
Then the grid should look like
.....
.....
.....
.....
..X..
```

7.1.2 Análise de Cobertura de Testes

Após a tradução de todos os cenários foi feita uma análise de cobertura de testes com base na abordagem textual e do *BehaviorMap*. Na abordagem BDD textual, e nos tutoriais das ferramentas atuais, é ensinado que as asserções são feitas apenas no passo *Then*, ou seja, apenas é verificado o resultado final, esquecendo a verificação do que permanece inalterado pela ação especificada no passo *When*. A abordagem do *BehaviorMap* cria assim asserções para o passo *Given* e *Then*, juntamente com asserções para verificar o que permanece inalterado pelo passo *When*. A Figura 7.3 mostra um gráfico com o número

de asserções feito com cada uma das abordagens. Pela Figura 7.3 consegue-se verificar que a abordagem do *BehaviorMap* assegura mais casos de testes, sem aumentar o esforço temporal dos utilizadores porque os testes são criados automaticamente.

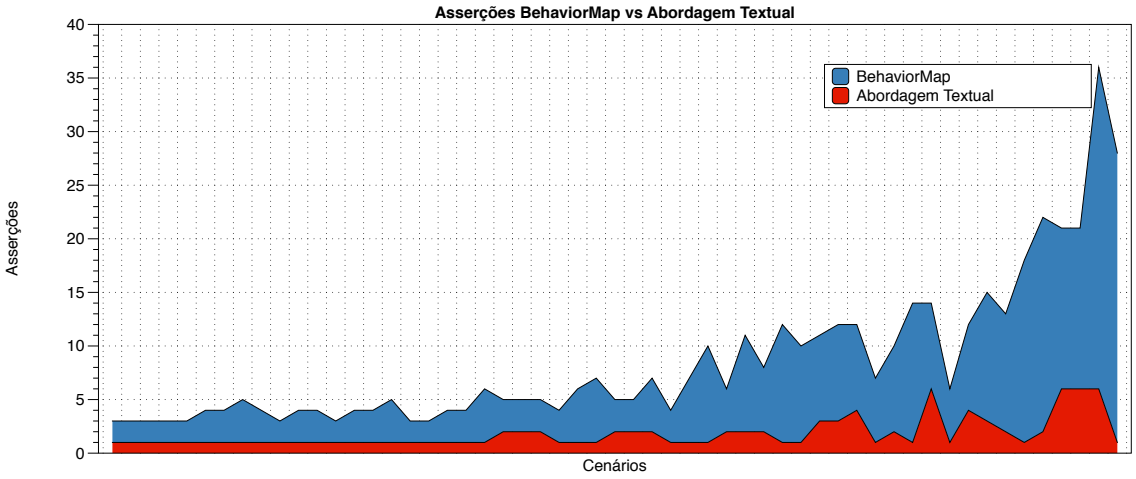


Figura 7.3: Asserções *BahaviorMap* vs Asserções abordagem Textual

7.2 Compreensão dos modelos

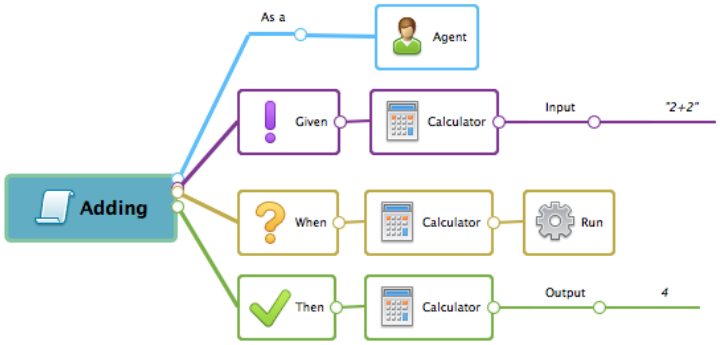
A compreensão dos modelos por parte de utilizadores comuns foi a segunda avaliação efetuada. A partir desta avaliação pretendíamos descobrir se os utilizadores tinham um menor esforço cognitivo ao serem confrontados com diagramas especificados pelo *BehaviorMap* em relação aos diagramas textuais. O esforço cognitivo é uma teoria que afirma que a capacidade da memória de trabalho duma pessoa é limitada e tem de ser utilizada com eficácia [Paa+03]. A carga total de esforço cognitivo é uma soma do esforço intrínseco (i.e. a dificuldade e complexidade da informação), do esforço *Germane*, que diz respeito ao esforço de aprendizagem e de memorização e do esforço *Extraneous* que está relacionado de como a informação é apresentada.

7.2.1 Desenho do Experimento

O experimento foi desenhado para testar a hipótese dos cenários desenvolvidos com o *BehaviorMap* serem mais facilmente compreendidos em relação aos cenários textuais. Para tentar responder a essa questão, tentámos replicar um experimento [Beg+14] que utilizou sensores biométricos para avaliar o esforço cognitivo ao analisar código C. O experimento realizado foi composto por: um treino, duas tarefas práticas e seis tarefas de compreensão. Cada participante realizou o experimento individualmente, acompanhado pelo autor da dissertação.

Treino

O treino tinha o objetivo de introduzir ao participante os elementos do experimento numa duração de 30 minutos, no máximo. Era explicado ao participante o que são cenários comportamentais, quais seriam os modelos utilizados no experimento, os seus constituintes e as ferramentas dos modelos. Inicialmente foi explicado ao participante que os cenários servem para representar comportamento e que vão ser utilizados dois diferentes tipos: textuais e gráficos. De seguida, foram mostrados dois exemplos de cenários, um escrito textualmente e outro graficamente para que o participante observasse as semelhanças e diferenças. Os cenários utilizados como exemplos no treino são apresentados nas Figuras 7.4, 7.5. Ao explicar os cenários gráficos era feita uma descrição do significado de cada nó e como ler um cenário. Após isso, eram explicadas as ferramentas que iriam ser utilizadas no experimento, o *JBehave* e o *BehaviorMap*. O participante fez os exemplos lecionados nas ferramentas com auxílio, caso precisasse, e era corrigido.



(a) Cenário gráfico

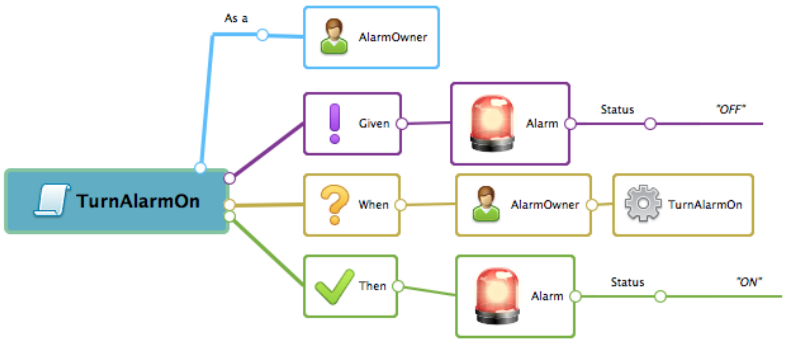
```
1 Feature: Adding
2 Scenario: Add two numbers
3
4 Given the input "2+2"
5 When the calculator is run
6 Then the output should be "4"
```

(b) Cenário textual

Figura 7.4: Cenário *Adding* gráfico e textual utilizado no treino

Tarefas práticas

As tarefas práticas tinham o objetivo de confrontar os participantes com o *BehaviorMap* e o *JBehave*. Eram sempre realizadas antes das tarefas de compreensão para que os participantes interagirem com as ferramentas. Nas tarefas práticas os participantes tinham de fazer uma tradução de um cenário textual para gráfico e vice-versa. Tinham sete minutos no total para realizar o experimento.



(a) Cenário gráfico

```
1  Feature: Alarm
2  Narrative:
3  In order to don't get robbed
4  As a AlarmOwner
5  I want to turn my alarm on
6
7  Scenario: Turn Alarm On
8  Given an alarm with status OFF
9  When I turn the alarm on
10 Then the Alarm status must be ON
```

(b) Cenário textual

Figura 7.5: Cenário *Turn Alarm On* gráfico e textual utilizado no treino

Tarefas de compreensão

As tarefas de compreensão serviram para avaliar se os participantes conseguiam compreender os modelos, respondendo a perguntas sobre os mesmos. Os participantes eram expostos ao cenário e a três perguntas (expostas no topo direito do ecrã). Os participantes tinham então de responder a essas três perguntas num espaço de cinco minutos. As perguntas utilizadas foram:

1. Quais as condições iniciais esperadas?;
2. Quais as ações a serem especificadas?;
3. Qual o resultado esperado?.

Na primeira questão, os participantes tinham de dizer o que estava especificado no ramo *Given*, na segunda tinham de dizer qual a ação ou ações especificadas no ramo *When*. Na terceira e última questão os participantes tinham de dizer o que estava especificado no ramo *Then*.

De forma a poder analisar as respostas dos participantes, era feita uma gravação de áudio durante a tarefa. Para além disso eram medidos os seus sinais biométricos (batimentos cardíacos e ondas cerebrais) para conseguir classificar melhor o esforço cognitivo exigido para realizar a tarefa.

7.2.2 Escolha dos cenários

A escolha dos cenários para realizar o experimento teve como base o conjunto de métricas criadas. As métricas foram desenhadas de forma a tentar diferenciar os cenários *BehaviorMap*, provenientes da tradução dos cenários textuais selecionados no estudo de aplicabilidade apresentado na Secção 7.1. As métricas concebidas foram:

1. Folhas (*TamanhoCenário*) - Contar número de folhas presentes num cenário no ramo *Given*, *When* e *Then* (Ignorando todos os nós que não sejam constantes em qualquer cenário (Título, *Given*, *When*, *Then* e Agente);
2. Ações no ramo When (*AçõesWhen*);
3. Ações no ramo Then (*AçõesThen*);
4. Entidades distintas (*EntidadesDistintas*).

As métricas são semelhantes às encontradas em diagramas UML, como as encontradas em diagramas de classes, de sequência ou de atividade [SDM14b]. Com a aplicação das métricas foram selecionados 8 cenários, 4 gráficos e 4 textuais. Os cenários foram agrupados em três complexidades, *Baixa*, *Média* e *Alta*.

Para a classe de complexidade *Alta* foram escolhidos os cenários que abrangiam mais conjuntos de valores máximos de todas as métricas aplicadas, para a classe *Baixa* foram

escolhidos os que tinham os mínimos e para a classe *Média*, os cenários que tinham os valores médios. A Tabela 7.1 apresenta os valores das métricas dos cenários selecionados. De forma a melhor visualizar os cenários recolhidos em relação à amostra total, a Figura 7.6 representa os valores recolhidos, num *box-plot*, onde são assinalados os cenários selecionados para o experimento de acordo com cada métrica.

Tabela 7.1: Resultado das métricas dos cenários selecionados

Descrição Cenário	Entidades	Nós Folha	Ações When	Ações Then
Textual Alta	8	28	1	1
Gráfico Alta	6	24	6	0
Textual Média	2	6	1	1
Gráfico Média	2	6	1	1
Textual Baixa	1	3	1	0
Gráfico Baixa	1	3	1	0
Prático Gráfico	2	5	1	2
Prático Textual	1	4	1	0

A Tabela 7.2 resume as médias e os respetivos desvios-padrão das métricas aplicadas nos cenários *BehaviorMap*. Arredondado todos os valores para baixo, a partir da tabela podemos verificar que cada cenário tem em média 6 folhas e o maior desvio padrão dos cenários recolhidos é em relação ao número de folhas do cenário, i.e., o seu tamanho. Além disso, os cenários encontrados especificam em média duas entidades, cada um realizava apenas uma ação por cenário (como é previsto pelas regras do BDD) e raramente é especificado uma acção no passo *Then*.

Tabela 7.2: Resumo dos resultados das métricas aplicadas nos cenários *BehaviorMap*

Métricas	Média	Desvio Padrão
NósFolha	6.00	4.869
AçõesWhen	1.13	0.708
AçõesThen	0.81	0.652
Entidades	2.49	1.648

A Figura 7.7, mostra os cenários para as tarefas práticas e as Figuras 7.8, 7.9, 7.10, 7.11 e 7.12 mostram cenários escolhidos para as tarefas de compreensão.

7.2.3 Captura de Informação

O experimento teve várias formas de captura de informação, utilizadas em situações distintas ou paralelamente. As fontes de informação utilizadas foram questionários, esforço temporal, respostas dadas e os sensores *MindWave* e *PulseSensor*.

Questionários

Foram utilizados três questionários. O primeiro questionário a ser utilizado tinha o propósito de obter informação sobre o perfil do participante. Qual o seu nome, idade, *background* e se já tinha alguma experiência com modelo comportamentais e BDD. O questionário é apresentado no B.1.

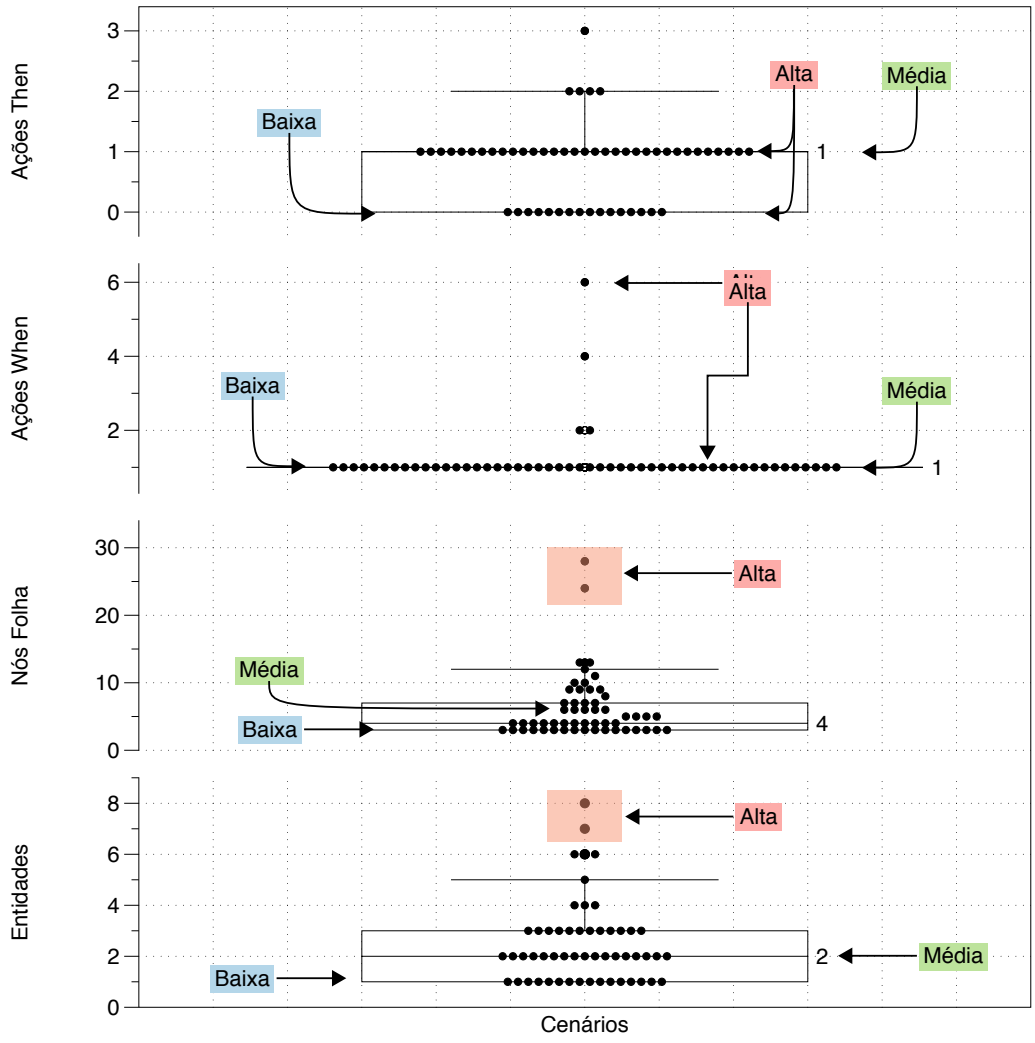
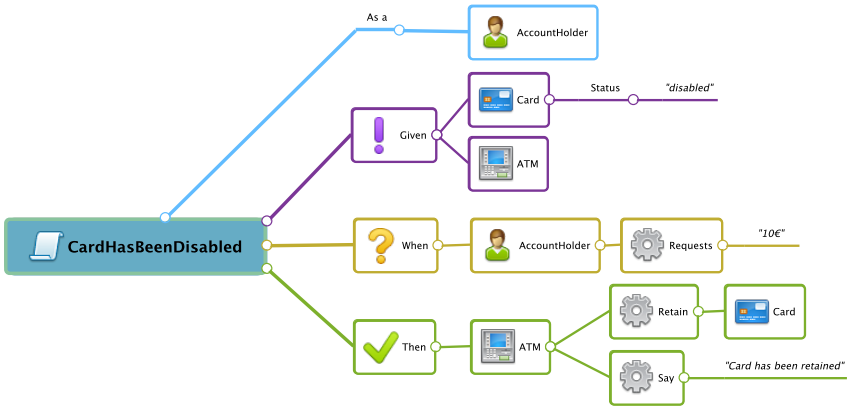


Figura 7.6: Distribuição das metricas na amostra



(a) Cenário gráfico - *TP1*

```
Scenario:  trader is not alerted below threshold

Given a stock of symbol STK1 and a threshold of 10.0
When the stock is traded at 5.0
Then the alert status should be OFF
```

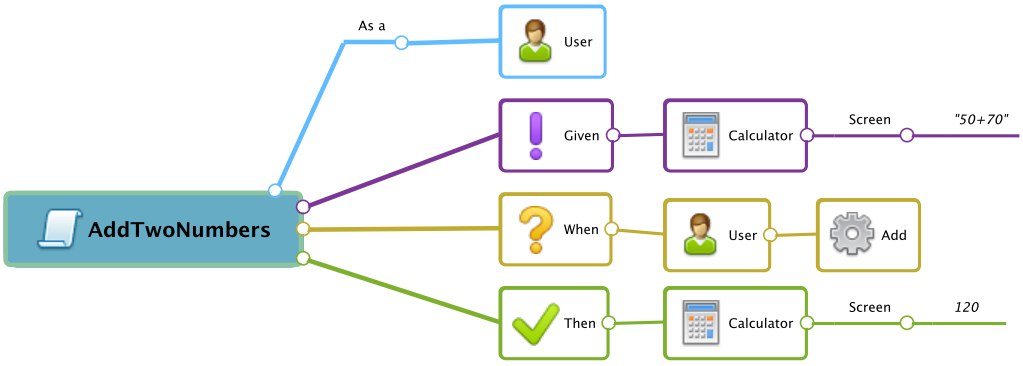
(b) Cenário textual - *TP2*

Figura 7.7: Cenário gráfico e textual para as tarefas práticas

```
Feature: Showtime Descriptions
Narrative:
In order to find movies that fit my schedule
As a movie goer
I want to see accurate and concise showtimes

Scenario: Hide minutes for times ending with 00
Given a movie
When I set the showtime to 2007-10-10 at 2:00pm
Then the showtime description should be "October 10, 2007 (2pm)"
```

(a) Cenário textual - *TC1*



(b) Cenário gráfico - *TC2*

Figura 7.8: Cenário gráfico e textual de complexidade baixa

```
Feature: Map View
Narrative:
In order to asses the situation in my area
As a coast guard
I want to see the location of each vessel marked on a map

Scenario: show vessel inside map area
Given vessel "Seal" at position "52.01N, 3.99E"
When I view the map area between "52.10N, 3.90E" and "51.90N, 4.10E"
Then I should see vessel "Seal" at position "52.01N, 3.99E"
```

Figura 7.9: Cenario textual de complexidade média - *TC3*

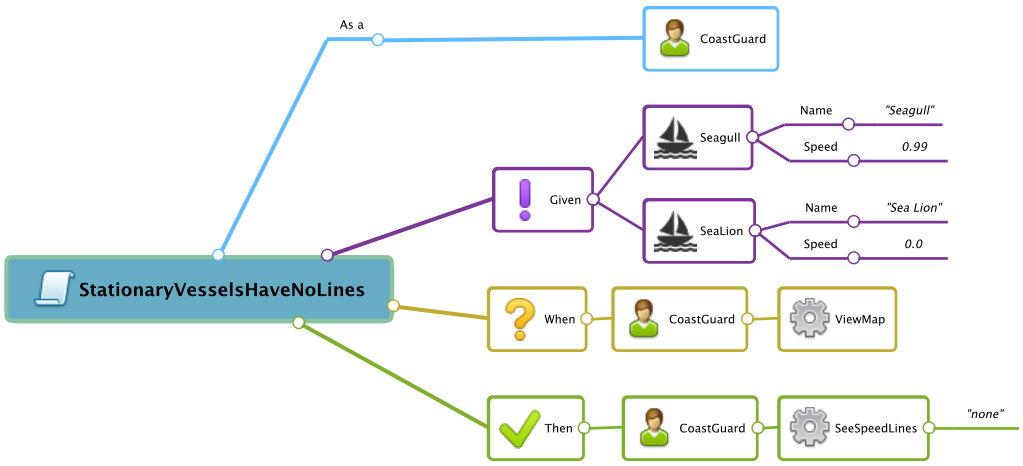


Figura 7.10: Cenario gráfico de complexidade média - *TC4*

```
Given a customer:
    |CustomerCode|CustomerName|Country|Currency|
    |6538764     |John Smith  |USA    |  USD   |
And a couple of portfolios :
    |CustomerCode|PortfolioCode|PortfolioType |
    |6538764     |FIFO_1       |FIFO          |
    |6538764     |AVG_1        |Average priced|
And a security:
    |SecurityID|SecurityName|Currency|Decimals|
    |NOK       |Nokia       |USD     |2       |
And some purchase lots in both portfolios :
    |TransactionType|SecurityID|Amount|Portfolio     |
    |Purchase       |NOK       |172.12|6538764/FIFO_1|
    |Sale           |NOK       |83.47 |6538764/FIFO_1|
    |Purchase       |NOK       |100.00|6538764/AVG_1 |
When the current position for customer "6538764" is requested
Then the system should display:
    |CustomerCode|PortfolioCode|SecurityID|Amount|
    |6538764     |FIFO_1       |NOK       |88.65 |
    |6538764     |AVG_1        |NOK       |100.00|
```

Figura 7.11: Cenario textual de complexidade alta - *TC5*

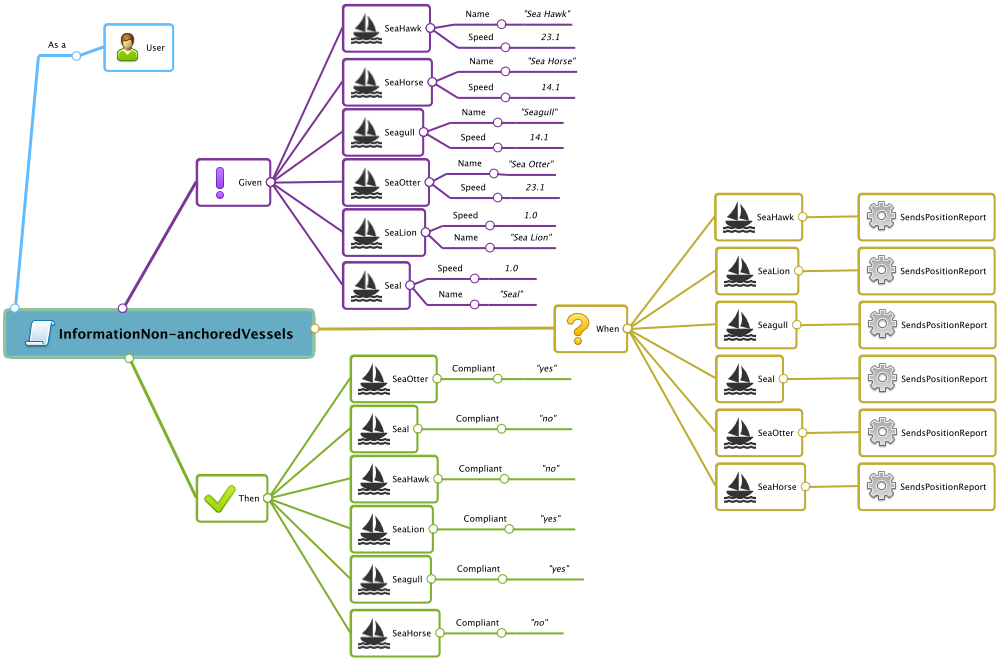


Figura 7.12: Cenario gráfico de complexidade alta - *TC6*

O segundo questionário serviu para capturar o esforço cognitivo dos participantes após a realização da tarefa. O questionário utilizado foi o NASA-TLX, um questionário bastante utilizado para avaliar subjetivamente o esforço cognitivo de uma pessoa na realização de tarefas [HS88]. O questionário era entregue aos participantes após a realização da tarefa e o participante fazia uma autoavaliação (de 0 a 100) em diversos domínios (*Performance*, Esforço Mental, Esforço Temporal, Esforço Físico, Frustração e Esforço) sobre a realização da tarefa. Cada domínio tinha um peso específico escolhido também pelo participante, fazendo comparações entre todos os domínios. O questionário pode ser visto no B.2.

O terceiro e último questionário serviu para fazer uma avaliação de todo o experimento por parte do participante. Se gostou do mesmo, se achou que a informação dada na aula foi adequada, classificação da dificuldade de cada tarefa numa escala de 1 a 10 e a possibilidade de dar uma opinião por escrito. O questionário pode ser visto no B.3.

Esforço temporal e Respostas

Para avaliar a dificuldade dos participantes ao realizar as tarefas foi medido o esforço temporal e as suas respostas. O tempo era iniciado com um cronómetro quando o participante iniciava a tarefa e encerrado quando terminava. As respostas no caso das tarefas de compreensão foram gravadas com um gravador de áudio para poderem ser posteriormente analisadas. Em relação às respostas das tarefas práticas, as mesmas foram posteriormente analisadas com base num conjunto de regras explicadas na secção 7.3.1.

MindWave e PulseSensor

Foram utilizados dois sensores biométricos para conseguir avaliar melhor a dificuldade cognitiva dos participantes a realizar as tarefas. Os sensores utilizados foram o *MindWave* e o *PulseSensor*. O *MindWave* é um dispositivo para medir a atividade elétrica cerebral (EEG). Um eletroencefalograma (EEG) mede a atividade cerebral que surge da ativação de neurónios [And00]. O *MindWave* é um aparelho que regista e envia sinais EEG a 512 Hz de uma localização única (testa do sujeito). O *MindWave* tem a capacidade de enviar sinais de diversas ondas cerebrais (*Gamma*, *Beta*, *Alpha*, etc) ou *Raw* e níveis de Atenção e Meditação. A Atenção e Meditação correspondem aos níveis de Atenção e Relaxamento [SDM14a]. São calculados pelo próprio aparelho. A aplicação desenvolvida para registar os sinais foi construída com base num *framework* gratuito, disponível no *site* do *MindWave* e desenhado pelo autor da dissertação.

O *PulseSensor* é um medidor de batimentos cardíacos. É um projeto *open-source* que funciona através do *Arduino*. O batimento cardíaco, tanto como a sua variação (HRV[1]), podem ser um medidor de esforço cognitivo [Fre+05; Wil02; Mul92]. Outros estudos [Muk+11; Hjo+04; Tae+11; Mat+12; Cin+13] mostraram que diferentes níveis de esforço mental têm efeitos no HRV, maior o esforço cognitivo, menor o HRV. A aplicação para registar os batimentos cardíacos utilizada é uma aplicação oficial do autor do *PulseSensor*, onde apenas foi-lhe adicionada a possibilidade de gravar os dados num ficheiro CSV. A Tabela 7.3 apresenta alguns dos efeitos conhecidos que se conseguem inferir a partir do EEG e dos BPM[2]. As aplicações de captura utilizadas no experimento são apresentadas na Figura 7.13.

Tabela 7.3: Efeitos cognitivos medidos através de EEG e BPM

Método	**Efeito**
Piscar de Olhos	Atenção Visual [De 90]; Níveis de *Stress* e ansiedade [Doe57]; Classificação do esforço visual durante voos [Wil02]; Esforço mental durante tarefas de controlo de trafego aéreo [BWS96]; Esforço mental durante tarefas aritméticas e visuais [RM05]
Análise de ondas (Alpha, Beta, Gamma, Delta, Theta)	Esforço mental durante tarefas de controlo de trafego aéreo [BWS96]; Esforço mental durante tarefas aritméticas e visuais [RM05]; Classificação cognitivas de tarefas [LT06]; Consciência auditiva [MJ96]
Rácios de ondas	*Memory load* durante tarefas cognitivas [Gri+08]; *Task engagement index* [Kra90; Law+01; Ber+00]; Estado de condutores em diversas condições [BW93]
Atenção e Meditação	Esforço cognitivo [Haa+10]
HRV	Esforço cognitivo

[1]HRV - Heart Rate Variance - Variação do batimento cardíaco
[2]BPM - Beats Per Minute - Batimentos por minuto

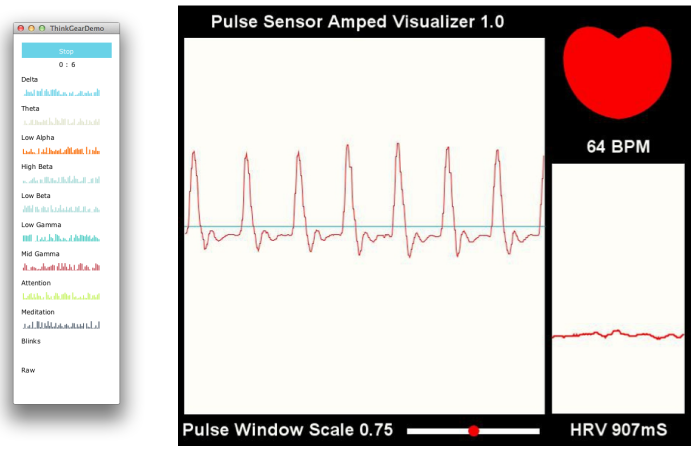


Figura 7.13: Ferramenta para captura dos sinais provenientes do *MindWave* e do *Pulse-Sensor*

7.2.4 Participantes

Os participantes neste experimento foram selecionados não tomando em consideração diferentes áreas de especialização. Esta decisão foi tomada porque os diagramas propostos para representar os cenários comportamentais foram construídos para conseguir abranger múltiplas especialidades, sem foco em nenhuma em particular. A única restrição para participar no experimento era o facto de não conhecerem e não dominarem a língua inglesa. No total participaram 15 pessoas no experimento, nove do sexo masculino e seis do sexo feminino. A Tabela 7.4 resume os dados dos participantes do experimento.

Tabela 7.4: Dados dos participantes do experimento

ID	Género	Background	Grau	Idade	Conhece modelos comportamentais?
01	Feminino	Medicina Veterinária	Mestre	23	Não
03	Feminino	Dietética e Nutrição	Licenciatura	24	Sim
06	Feminino	Communication Science	Mestre	32	Não
08	Feminino	Medicina Veterinária	Pos-Graduada	23	Sim
09	Feminino	Nenhum	Nenhum	23	Não
14	Feminino	Biologia	Licenciatura	23	Não
02	Masculino	Nenhum	Nenhum	55	Não
04	Masculino	Eng. Informática	Licenciatura	23	Sim
05	Masculino	Eng. Informática	Licenciatura	23	Sim
07	Masculino	Eng. Informática	Licenciatura	23	Sim
10	Masculino	Electrical Engineering	Licenciatura	20	Não
11	Masculino	Electrical Engineering	Licenciatura	28	Sim
12	Masculino	Teologia	Licenciatura	55	Sim
13	Masculino	Nenhum	Nenhum	41	Não
15	Masculino	Nenhum	Nenhum	20	Não

A Figura 7.14 resume os dados dos participantes em relação ao seu curso e grau obtido. Quatro dos participantes não tinham nenhuma área de especialização. Três deles pertenciam às ciências da computação e dois deles à engenharia eletrónica. Quatro participantes estão agrupados nas ciências da natureza (Medicina Veterinária, Nutrição e

Biologia). Uma pessoa pertencia à área de comunicação social. A maioria dos participantes era licenciada, onde existiam dois mestres e um pós-graduado. Quem não tinha grau de especialização foram os participantes que não estudaram nenhuma área em particular, restando um que está de momento a realizar uma licenciatura. Nenhum dos participantes conhecia o BDD antes de realizar a experiência e apenas oito dos participantes conheciam modelos comportamentais.

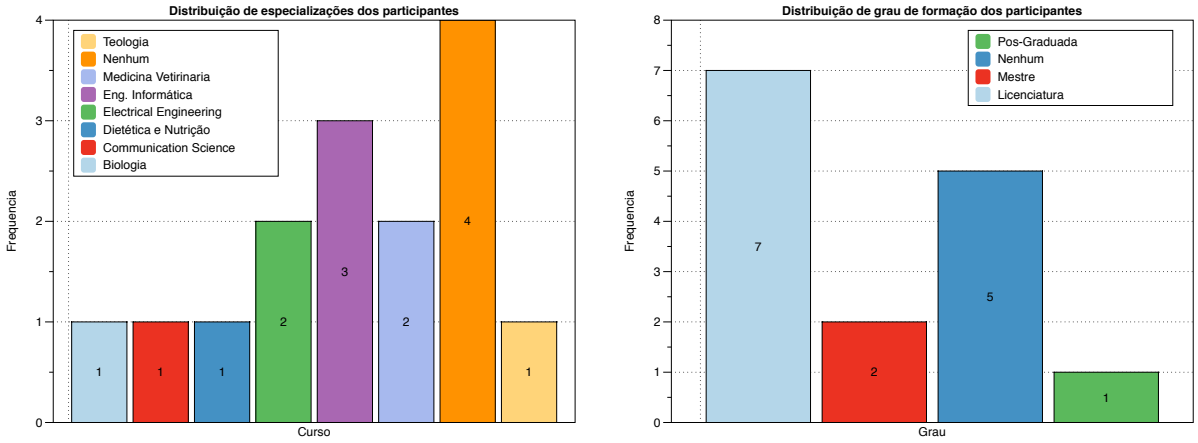


Figura 7.14: Distribuição das especializações e graus de formação dos participantes

7.2.5 Ambiente de trabalho

O experimento foi realizado em diversos locais, dependendo da localização do participante. Todos os experimentos foram realizados no mesmo computador, um portátil com a utilização de um rato *wireless*. A distância do portátil para o participante era ajustada dependendo do mesmo, até que estivesse do seu agrado visualmente. Relativamente aos sensores, o *PulseSensor* foi instalado na mão que não era utilizada para escrita, de forma a não ter de ser removido na altura de preenchimento de questionário. O *MindWave* foi colocado nos participantes, ajustando a largura do mesmo para não causar nenhum incomodo. Os sensores biométricos eram só instalados no início das tarefas de compreensão. A Figura 7.15 ilustra um indivíduo utilizando o *MindWave* e o *Pulse Sensor*.

7.2.6 Processo

O processo de realização do experimento está demonstrado na Figura 7.16. A primeira atividade, *Explicar o experimento*, consistia em explicar ao participante como iria ser executado o experimento e qual o seu papel no mesmo e no preenchimento de um questionário sobre os dados do participante. De seguida, a segunda tarefa, *Aula sobre BDD*, servia para explicar ao participante do experimento os conceitos do BDD.

Após esta parte introdutória, era iniciada a execução das tarefas práticas, atividade *Iniciar tarefas práticas*. Antes do início das tarefas era explicado ao participante que o

Figura 7.15: Indivíduo utilizando o *MindWave* e o *Pulse Sensor*

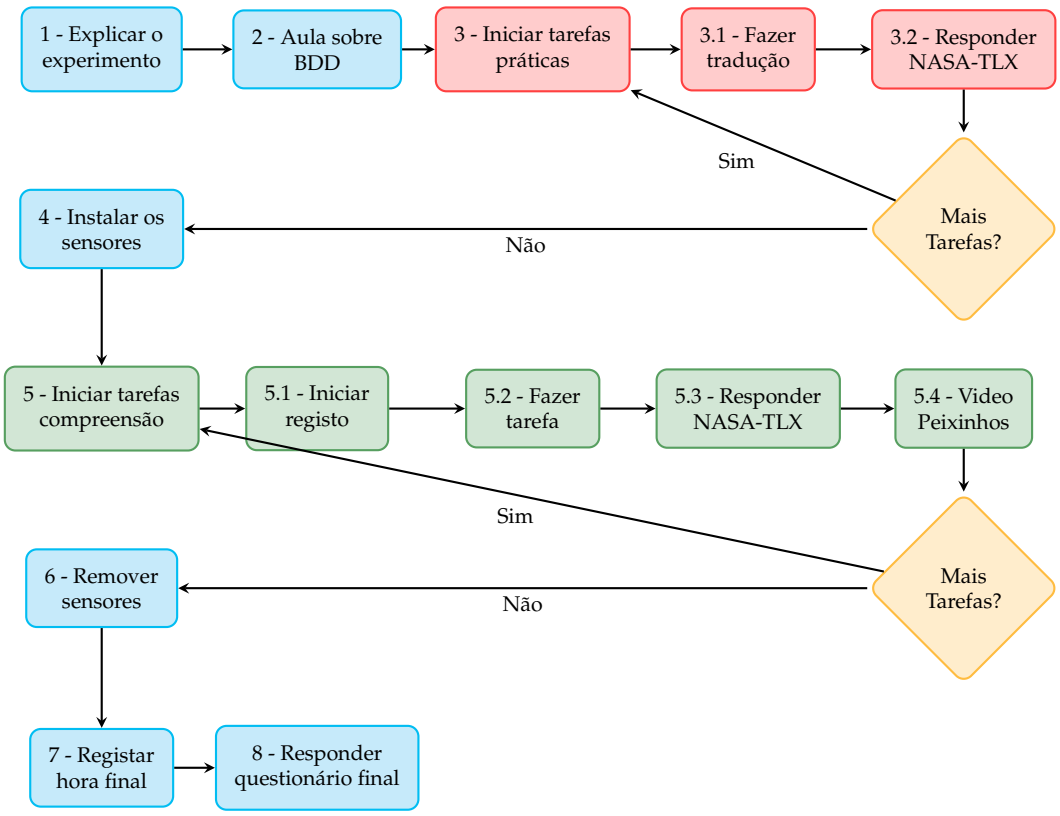


Figura 7.16: Processo do Experimento

objetivo das tarefas práticas seria realizar uma tradução de um cenário textual para gráfico e vice-versa. Também era introduzido ao participante o questionário *NASA-TLX*, questionário que iria responder no final de cada tarefa. De seguida, o participante executava a tarefa *Fazer tradução*, onde era iniciado um cronómetro para contar a duração da tarefa. Após terminar a tarefa, era parado o cronómetro e o participante executava a atividade *Responder NASA-TLX*. Qualquer dúvida levantada por parte do participante sobre o questionário era esclarecida e tinha tempo ilimitado para responder ao mesmo. Caso existissem mais tarefas práticas a executar o processo repetir-se-ia a partir da atividade 3.

Caso contrário, o processo seguia para as tarefas de compreensão. Antes de iniciar as tarefas de compreensão era necessário *Instalar os sensores MindWave* e *PulseSensor*. Após isso era iniciada a atividade *Iniciar tarefas de compreensão*. As tarefas eram realizadas por ordem aleatória de complexidade, i.e., ao realizar uma tarefa de complexidade baixa, por exemplo, com um cenário textual a seguinte tarefa a executar seria uma com a mesma complexidade mas com um cenário gráfico. Por cada tarefa era realizada a atividade *Iniciar registo* que consistia em iniciar o registo dos sensores e de um gravador para ouvir as respostas do participante. Caso alguma resposta dum participante fosse incompreensível era pedido para que repetisse. O participante executava a tarefa, a captura dos sensores era terminada e o participante respondia ao questionário NASA-TLX. Após responder ao questionário, o participante visualizava o vídeo duns peixinhos com os sensores do *MindWave* e *PulseSensor* ativos. O propósito desta atividade era que o participante relaxasse de tarefa para tarefa. Isto foi necessário porque de acordo com [Beg+14] o esforço mental dos participantes aumentava gradualmente de tarefa para tarefa caso não houvesse interrupções. Estas atividades eram repetidas por cada tarefa de compreensão.

Após todas as tarefas de compreensão serem executadas, os sensores eram removidos, era registada a hora e o participante respondia ao questionário final. Após responder ao questionário eram feitas eventuais perguntas ao participante sobre o que achou do experimento e qual a forma que preferiu para representar os cenários.

7.3 Método de Análise

Os dados capturados foram comparados utilizando uma análise de variância não paramétrica *Kruskall-Wallis* [KW52] e *Mann-Whitney* [MW47]. A escolha de análises não paramétricas, deveu-se ao facto dos dados não possuírem uma distribuição normal. Foi realizado o teste *Anderson-Darling* [AD52] e foi concluído que não apresentavam uma distribuição normal com um grau de confiança de 99%. A análise de variância testa se as diferenças entre as médias de grupos são causadas por fatores aleatórios. A análise de variância considera duas hipóteses:

1. **Hipótese 0**: A diferença de médias dos grupos é causada por fatores aleatórios, i.e, as médias são consideradas iguais, não existem diferenças significativas.

2. **Hipótese 1**: A diferença de médias dos grupos **não** é causada por fatores aleatórios, i.e, as médias são consideradas **diferentes**, existem diferenças significativas.

As análises de variância foram realizadas para considerar dois fatores, a forma como o cenário BDD foi escrito (textualmente ou graficamente) e a sua classe de complexidade, para verificar se existem diferenças significativas com a mudança de complexidade. Os resultados das análises realizadas foram obtidos com um grau de confiança de 95%.

7.3.1 Avalição das respostas dos participantes

Nas tarefas práticas, a avaliação das traduções foi feita da seguinte forma: em cada tradução esperava-se que os elementos do modelo inicial estivessem capturados no modelo final. Então, na tradução do mapa mental para texto esperava-se que os elementos capturados estivessem no texto final. Por exemplo a Listagem 7.5, mostra uma tradução do cenário gráfico (Figura 7.7), criada por um participante do experimento. Para avaliar a tradução foi verificado: se o título correspondia ao original; se no passo *Given* eram mencionados um *Cartão* com estado *desativado* e um *ATM*; No passo *When* se o agente *AccountHolder* cometia a ação *levantar* com parâmetro *10*; No passo *Then* verificava-se se o ATM cometia a ação *Reter* o *Cartão* e dizia *Cartão foi removido*.

Listagem 7.5: Tradução de um participante de um cenário gráfico para textual

```
Scenario: Card Has Been Disabled

Given a card that is disabled and an ATM
When the AccountHolder Requests 10
Then the ATM Retains the Card and Says "Card has been removed"
```

Relativamente à tradução do texto (Figura 7.7) para mapa mental, pretendia-se que o título do cenário fosse *Trader is not alerted bellow threshold* e no ramo *Given* estivesse uma entidade *Stock* com os atributos *Threshold* e *Symbol* com os valores *10.0* e *STK1* respetivamente. No ramo *When* a entidade *Stock* executava a ação *Traded* com o valor *5*. Finalmente no ramo do *Then*, a entidade *Stock* devia ter o atributo *AlertStatus* com o valor *OFF*.

No final, a avaliação da tradução de cada participante foi feita através de uma nota. A nota foi calculada através do somatório dos elementos corretos presentes nos cenários.

Já a avaliação das tarefas de compreensão foi feita da seguinte forma: cada resposta dada pode ter resultados possíveis: correto, incompleto ou incorreto. Correto é quando um participante responde a tudo o que é suposto, incompleto é quando não responde a tudo e errado é quando a resposta está completamente errada. A secção 7.2.1 explica as respostas esperadas dos participantes.

7.4 Resultados da experiência

O experimento tinha o objetivo de fazer uma comparação entre diagramas comportamentais textuais e gráficos em relação à sua compreensão por parte dos seus utilizadores. A

compreensão foi avaliada utilizando cinco parâmetros:

1. Perguntas sobre o modelo;
2. Esforço temporal para responder;
3. Questionário sobre a dificuldade cognitiva;
4. Medição dos níveis de *Atenção* e *Meditação* através do *MindWave*;
5. Medição do *HRV* através do *PulserSensor*.

7.4.1 Tarefas Práticas

Nas tarefas práticas, o objetivo era efetuar uma tradução de diagrama textual para gráfico e vice-versa. A Tabela 7.5 apresenta as médias dos resultados obtidos de cada característica (*workload*, tempo e respostas certas) em relação às tarefas práticas. A Figura 7.18 apresenta um *box-plot* com a distribuição dos resultados para cada uma das tarefas. A tarefa *TP1* (tradução de um mapa mental para texto) teve uma média de *workload* menor em relação à tarefa *TP2* (tradução de texto para mapa mental), uma duração temporal em média inferior por aproximadamente 2 minutos e em relação às respostas certas, 90% em média face a 50%, uma diferença de 40%.

As análises de variância das tarefas práticas assumem que a variação é no cenário origem que é oferecido e não no uso da ferramenta utilizada.

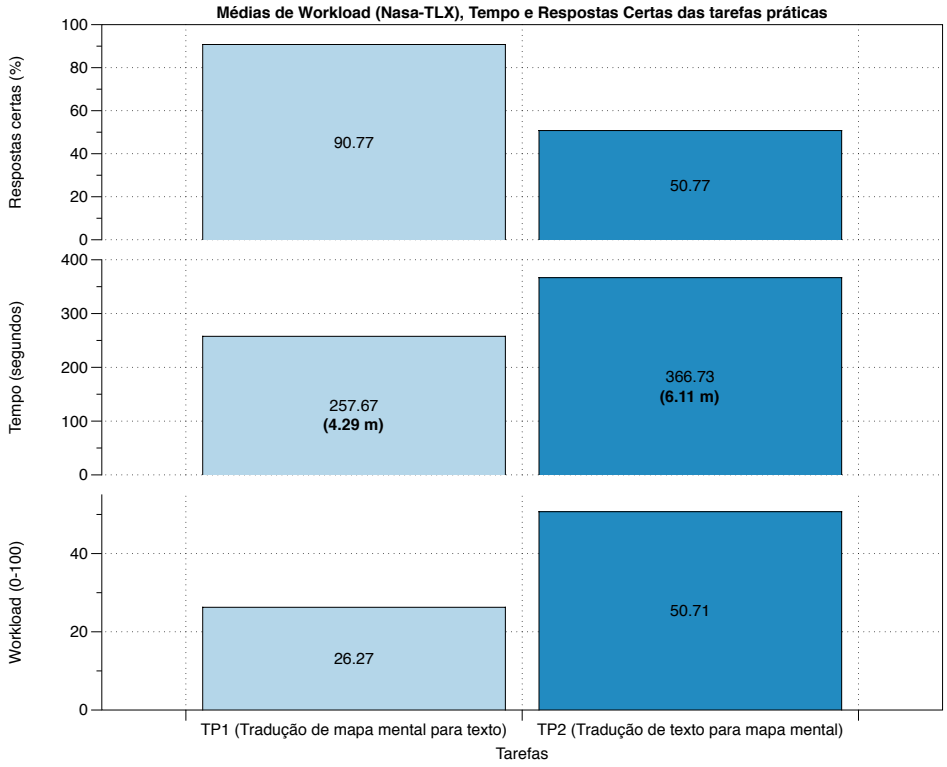


Figura 7.17: Médias de Workload (NASA-TLX), Tempo de Execução e Respostas Certas das tarefas práticas

Tabela 7.5: Resumo do Workload, Tempo de Execução e Repostas certas para tarefas práticas

Medição	Tarefas			
	TP1		TP2	
		s		s
***Workload* (0-100)**	26.26	18.17	50.70	25.00
Tempo (m)	4.29	1.31	6.11	1.13
Respostas certas (%)	90.77	17.00	50.77	21.77

Os participantes tiveram mais facilidade a executar a tarefa *TP1* em relação à *TP2* de acordo com o tempo de execução, o *workload* obtido e a percentagem de traduções corretas. Em relação ao *workload*, na tarefa *TP1* a maioria dos resultados (75%) têm valores até aos 35, face à tarefa *TP2* que apresenta valores até aos 70. Em relação ao tempo de realização da tarefa, 75% do tempo registado é abaixo dos 5 minutos para a tarefa *TP1* em comparação com os 7 minutos (tempo limite) para a tarefa *TP2*. Finalmente, em relação à conformidade da tradução feita, para a tarefa *TP1* existe um caso de *outlier* com 33% de respostas certas, onde os restantes participantes têm 92% ou 100% de nota da tradução. Na tarefa *TP2*, 75% dos participantes teve notas abaixo dos 60%, onde se registou quase um 100% por parte de um participante.

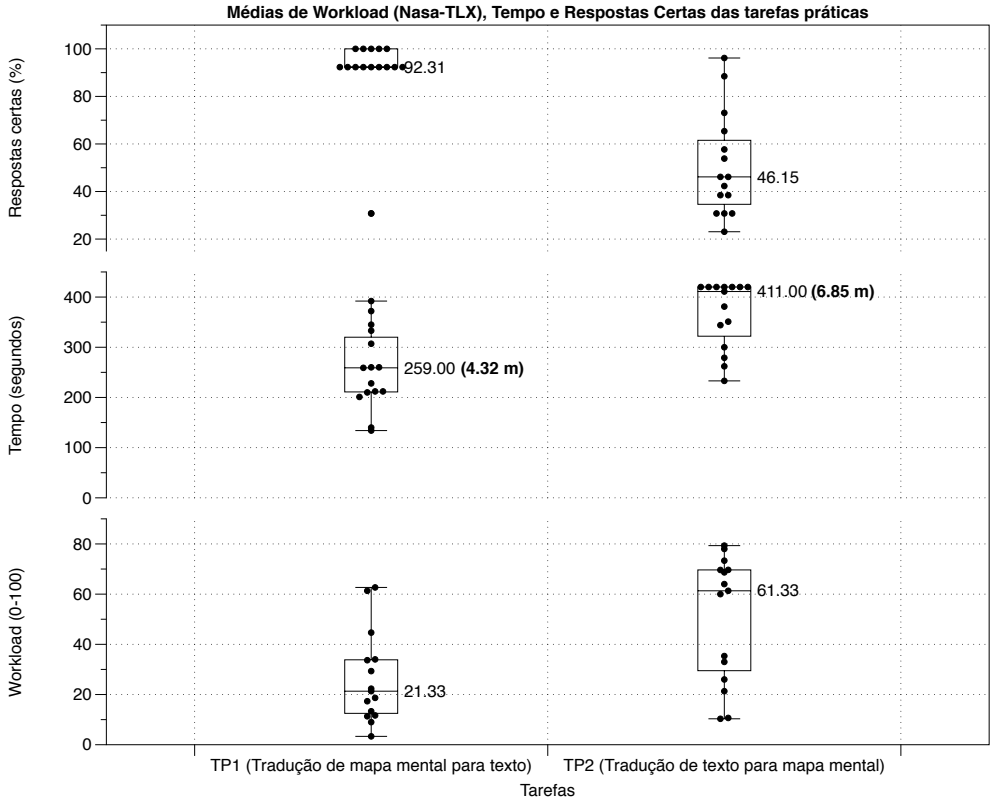


Figura 7.18: Workload, Tempo de Execução e Respostas Certas das tarefas práticas

Dificuldade cognitiva

De forma a verificar se as diferenças entre as médias obtidas não foram causadas por acontecimentos aleatórios, foi aplicada uma análise de variância (*Mann-Whitney*). O resultado (U = 174.0, p = 0.01) mostrou que a diferença média de *workload* entre as tarefas não foi causada por acontecimentos aleatórios, mas sim pela diferença entre o tipo de modelos. Este resultado foi obtido com um grau de confiança de 95%.

Esforço temporal e traduções

Em relação ao esforço temporal e as traduções realizadas foi também utilizada uma análise de variância (*Mann-Whitney*). O resultado da análise mostrou que a diferença entre os valores de esforço temporal (U = 196.0, p = 0.00) e traduções (U = 23.0, p = 0.00) não foram causados por fatores aleatórios, mas sim pela diferença entre o tipo de modelos. Estes resultados foram obtidos com um grau de confiança de 95%.

7.4.2 Tarefas de compreensão

Nas tarefas de compreensão, os participantes tinham de responder a três perguntas para cada modelo, num espaço de tempo de cinco minutos. Foram mostrados seis modelos no total, existindo três modelos gráficos e três textuais. Cada modelo apresentando pertencia a uma classe de complexidade (Baixa, Média, Alta), detalhada na secção 7.2.2. Os resultados médios obtidos na realização das tarefas de compreensão são apresentados na Figura 7.19. A Tabela 7.6 apresenta em detalhe as médias e desvios padrão das características das tarefas gráficas e textuais. De acordo com os dados registados, as tarefas utilizando modelos comportamentais gráficos tiveram um esforço e tempo de execução menor e obtiveram mais respostas certas.

Tabela 7.6: Resumo das medições para tarefas de compreensão, gráficas e textuais

Medição	Tarefas			
	Gráficas		Textuais	
		s		s
***Workload* (0-100)**	24.68	21.75	34.54	28.17
Tempo (m)	1.64	1.18	2.09	1.29
Respostas certas (%)	96.27	16.30	84.36	31.53

Para determinar quais foram as tarefas mais complicadas para os participantes, foi realizada uma análise para cada uma das tarefas realizadas. A Figura 7.20 mostra as tarefas realizadas, ordenadas por complexidade (Baixa, Média e Alta) começando por uma tarefa com diagrama textual intercalando com uma tarefa com diagrama gráfico. A Tabela 7.7 resume, para cada tarefa de compreensão, as médias para cada medição e o seu respectivo desvio padrão.

Os resultados mostram que maior parte das tarefas registaram um *workload* inferior a 50, excetuando a tarefa *TC5* que teve uma mediana de 68.7. A tarefa *TC5* foi a que registou a menor percentagem de respostas corretas, o maior tempo de execução e *workload*, logo

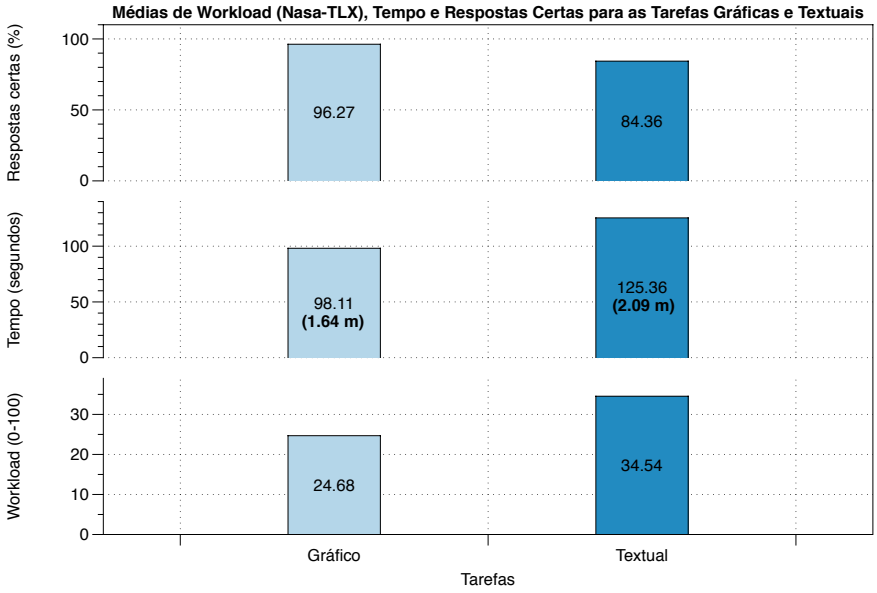


Figura 7.19: Workload (NASA-TLX), Tempo de Execução e Respostas Certas médios para tarefas gráficas e textuais

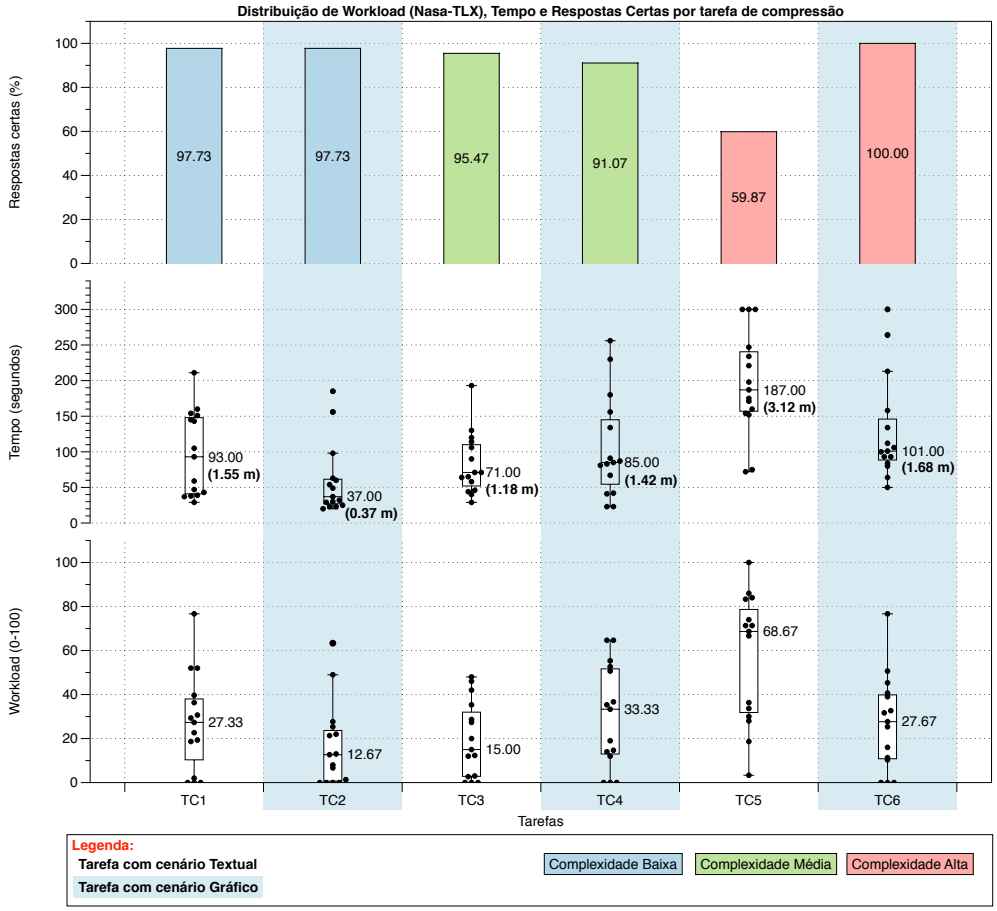


Figura 7.20: Workload (NASA-TLX), Tempo de Execução e Respostas Certas das tarefas de compreensão

Tabela 7.7: Resumo das medições para tarefas de compreensão

Medição	Tarefas											
	Complexidade *Baixa*				Complexidade *Média*				Complexidade *Alta*			
	TC1		TC2		TC3		TC4		TC5		TC6	
		s		s		s		s		s		s
***Workload* (0-100)**	27.11	22.27	16.69	18.94	19.49	17.41	39.20	23.46	57.02	29.19	27.16	21.68
Tempo (m)	1.62	0.99	0.98	0.84	1.38	0.72	1.75	1.20	3.27	1.21	2.17	1.22
Respostas certas (%)	97.73	8.78	97.73	8.78	95.74	11.96	91.07	26.67	59.87	42.02	100.00	0
TAREFAS GRÁFICAS												

considerada a tarefa mais complicada para os participantes. Em relação ao tempo de execução, a mediana dos tempos de execução é inferior a *1.68 minutos*, exceto para a tarefa *TC5*. As respostas certas são todas superiores a 90%, exceto também para a tarefa *TC5* com 59.87%.

A tarefa *TC5* é uma tarefa textual e de complexidade alta, apresentada na Figura 7.11. Representa o comportamento que o sistema deve obedecer ao requisitar o código do cliente *6538764*. Dado um conjunto de entidades *customer*, *portefolios*, *security* e *purchase lots*, ao requisitar o código de cliente *6538764*, o sistema deve apresentar duas entradas de junções de atributos de diferentes entidades (semelhante a um *join* em SQL). O cenário foi retirado de uma dissertação que utilizou o cenário como um típico caso em que o BDD não é facilmente percebido [Kou01]. Esta tarefa também foi considerada a mais difícil para os participantes com um resultado de *6.9* numa escala de 1 a 10, sendo o 1 muito fácil e o 10 muito difícil. A razão pela qual isto aconteceu não pode ser totalmente conhecida. A tarefa é considerada mais complicada pelas métricas propostas, mas não conseguimos averiguar se tivesse sido apresentada graficamente, se resultaria em resultados mais favoráveis.

Dificuldade cognitiva

De modo a verificar se as diferenças em relação ao esforço cognitivo entre os cenários gráficos e os cenários textuais são significativas, foi utilizada uma análise da variância (*Mann-Whitney* e *Kruskall-Wallis*). Realizou-se uma análise de variância fixando os tipos de cenários e variando as complexidades e fixando a complexidade variando os tipos. A Figura 7.21 mostra os valores médios registados de esforço cognitivo para cada tarefa.

A análise de variância (*Kruskall-Wallis*) mostrou que os cenários gráficos não tiveram uma diferença relevante entre eles de acordo com as mudanças de complexidades, com um grau de confiança de 95%. Relativamente aos cenários textuais, o mesmo não aconteceu. Houve uma diferença relevante dos cenários de complexidade *Baixa* e *Média* em relação aos cenários de complexidade *Alta*. A Tabela 7.8 apresenta os cálculos efetuados.

Para determinar qual é a média que difere na análise de *workload* nas tarefas de texto foi utilizada a análise *Mann-Whitney*. Os resultados são apresentados na Tabela 7.9. Com a análise podemos concluir que as tarefas de complexidade *Alta* são as que afetam as

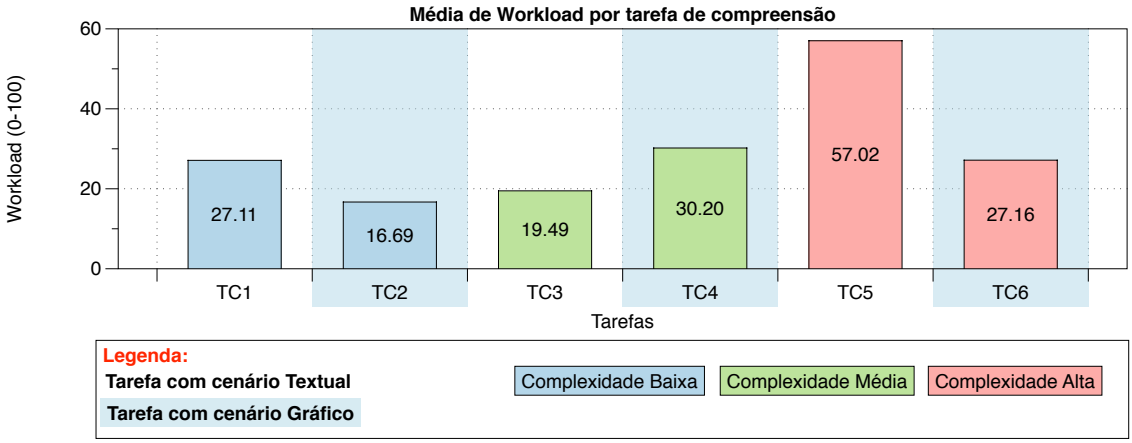


Figura 7.21: Médias de Workload por tarefa de compreensão

Tabela 7.8: Resumo análise *Kruskall-Wallis* - Fixando tipo de cenário e variando a complexidade

Comparação (classes de complexidade)	H, p	Conclusão ($\alpha = 0.05$)
Tarefas Textuais	12.48, 0.0019	**Existe uma relação entre as mudanças de complexidade e o *Workload*.**
Tarefas Gráficas	3.25, 0.1969	Não existe relação entre mudanças de complexidade e o *Workload*.

diferenças em relação ao *workload* nas tarefas textuais.

Tabela 7.9: Resumo análise *Kruskall-Wallis post hoc* com *Mann-Whitney* para cenários textuais

Comparação (classes de complexidade)	U, p	Conclusão ($\alpha = 0.05$)
Baixa vs Média	133.0, 0.40	Não existe diferença significativa.
Média vs Alta	192.0, 0.00	**Existe diferença significativa**.
Baixa vs Alta	49.0, 0.01	**Existe diferença significativa**.

Na análise, fixando a classe de complexidade, variando o tipo de cenário e utilizando a técnica *Mann-Whitney*, verificou-se que nas classes de complexidade *Baixa* e *Média*, as diferenças registadas em relação ao tipo de cenário não são significativas. No entanto, para os cenários de complexidade *Alta* o mesmo não aconteceu. A análise mostrou que o facto de ser gráfico ou textual tem consequências no *workload* registado. Os resultados são apresentados na Tabela 7.10.

Esforço temporal

De modo a verificar se as diferenças em relação ao esforço temporal entre os cenários gráficos e os cenários textuais são significativas, foi utilizada uma análise da variância. Realizou-se uma análise de variância, fixando os tipos de cenários, variando as complexidades e fixando a complexidade, variando os tipos. A Figura 7.22 mostra os valores médios registados de esforço temporal para cada tarefa. Em relação à análise de variação

Tabela 7.10: Resumo análise *Mann-Whitney* - Fixando a complexidade e variando o tipo de cenário

Comparação (tipo de cenário)	U, p	Conclusão ($\alpha = 0.05$)
Baixa	78.0, 0.15	Não existe relação entre o *Workload* e o tipo de cenário.
Média	141.5, 0.23	Não existe relação entre o *Workload* e o tipo de cenário.
Alta	50.0, 0.01	**Existe relação entre o *Workload* e o tipo de cenário**.

de complexidades fixando o tipo de cenário foi verificado que a complexidade influencia o esforço temporal para ambos os tipos de cenário. A Tabela 7.11 apresenta os cálculos efetuados.

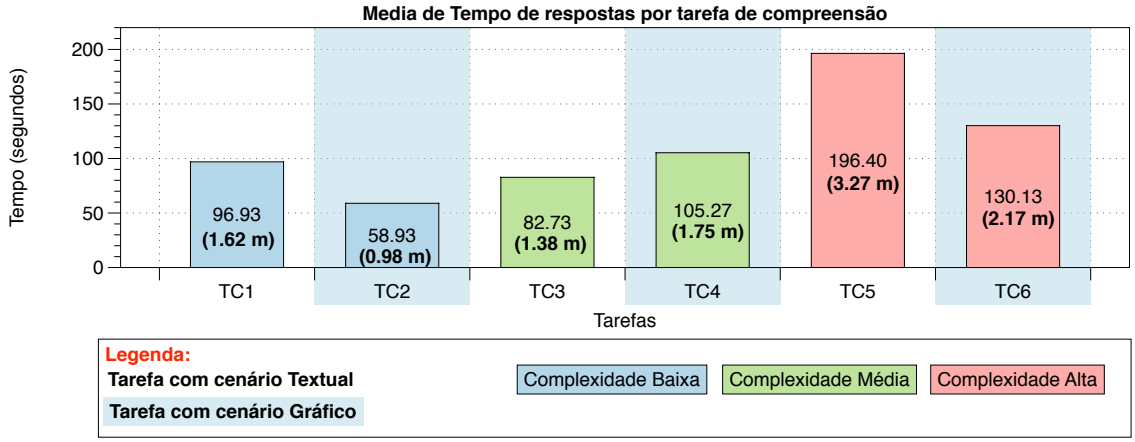


Figura 7.22: Médias de tempo por tarefa de compreensão

Tabela 7.11: Resumo análise *Kruskall-Wallis* - Fixando tipo de cenário e variando a complexidade

Comparação (classes de complexidade)	H, p	Conclusão ($\alpha = 0.05$)
Tarefas Textuais	18.88, < 0.0001	**Existe uma relação entre as mudanças de complexidade e o Esforço Temporal**.
Tarefas Gráficas	11.67, 0.0029	**Existe uma relação entre mudanças de complexidade e o Esforço Temporal.**

A fim de determinar qual a média de esforço temporal que difere em ambos os tipos de cenário, foi aplicada a análise *Mann-Whitney*. A Tabela 7.12 apresenta os cálculos efetuados para os cenários textuais. Podemos concluir que as tarefas de complexidade *Alta* afetam o esforço temporal em relação às duas outras classes de complexidade.

A Tabela 7.13 apresenta os cálculos efetuados para os cenários gráficos. Na comparação da classe *Baixa* com a *Alta* e com a classe *Média*, foram registadas diferenças significativas. Com isto podemos afirmar que os resultados do esforço temporal na classe de complexidade *Baixa* foram muito baixos em relação aos outros dois grupos, o que neste caso, é vantajoso. Os resultados são apresentados com 95% de grau de confiança.

Na análise de variância (*Mann-Whitney*) fixando a complexidade e variando o tipo de

Tabela 7.12: Resumo análise *Kruskall-Wallis post hoc* com *Mann-Whitney* para cenários textuais

Comparação (classes de complexidade)	U, p	Conclusão ($\alpha = 0.05$)
Baixa vs Média	105.5, 0.77	Não existe diferença significativa.
Média vs Alta	18.0, 0.00	**Existe diferença significativa**.
Baixa vs Alta	198.0, 0.00	**Existe diferença significativa**.

Tabela 7.13: Resumo análise *Kruskall-Wallis post hoc* com *Mann-Whitney* para cenários gráficos

Comparação (classes de complexidade)	U, p	Conclusão ($\alpha = 0.05$)
Baixa vs Média	162.5, 0.04	**Existe diferença significativa**.
Média vs Alta	145.5, 0.17	Não existe diferença significativa.
Baixa vs Alta	193.0, 0.00	**Existe diferença significativa**.

cenários, os resultados mostraram que na classe de complexidade *Baixa* e *Alta* as diferenças de médias não foram provocadas por fatores aleatórios, mas sim, pela diferença no tipo de cenários. Relativamente aos cenários de complexidade média, os resultados mostraram que não existiram diferenças significativas. A Tabela 7.14 apresenta os resultados apurados. Todos os resultados apresentados foram obtidos um grau de confiança de 95%.

Tabela 7.14: Resumo análise *Mann-Whitney* - Fixando a complexidade e variando o tipo de cenário

Comparação (tipo de cenário)	U, p	Conclusão ($\alpha = 0.05$)
Baixa	65.0, 0.05	**Existe relação entre o Esforço Temporal e o tipo de cenário**.
Média	129.0, 0.49	Não existe relação entre o Esforço Temporal e o tipo de cenário.
Alta	56.5, 0.02	**Existe relação entre o Esforço Temporal e o tipo de cenário**.

Respostas

Para verificar se as diferenças registadas em relação ao número de respostas corretas entre os cenários gráficos e os cenários textuais são significativas, uma análise da variância foi utilizada. Realizou-se duas análises de variância, uma fixando os tipos de cenários e variando as complexidades e outra fixando a complexidade e variando os tipos. A Figura 7.23 mostra os valores médios registados de esforço temporal para cada tarefa.

A análise de variância (*Kruskall-Wallis*) mostrou que em relação as cenários gráficos não existem diferenças na média de respostas certas por diferentes complexidades. Já nos cenários textuais existem diferenças em diferentes grupos de complexidades. A Tabela 7.15 apresenta os cálculos efetuados. De forma a analisar qual a classe de complexidade que gerou a diferença na análise de variância, foi aplicada a análise *Mann-Whitney*. A tabela 7.16 mostra os cálculos efetuados. Podemos concluir que a classe de complexidade *Alta* afetou as diferenças de médias existentes.

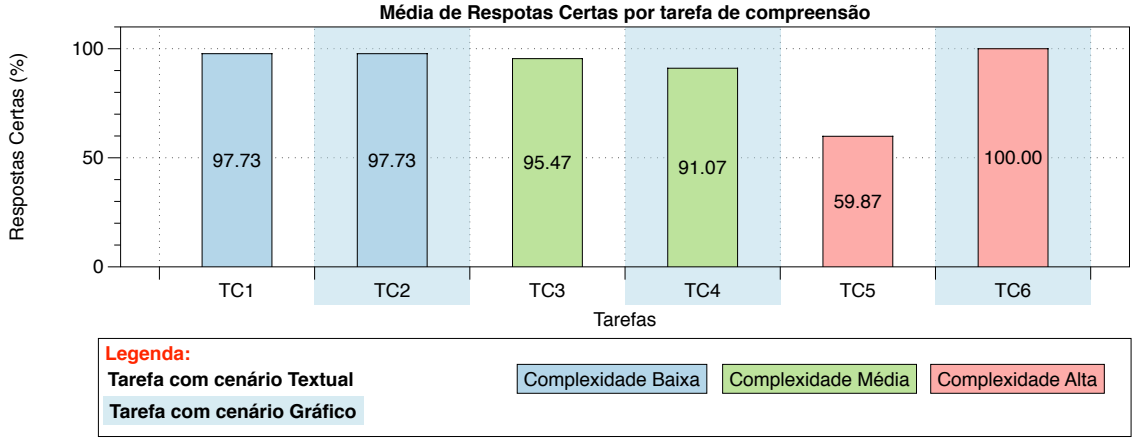


Figura 7.23: Médias de respostas certas por tarefa de compreensão

Tabela 7.15: Resumo análise *Kruskall-Wallis* - Fixando tipo de cenário e variando a complexidade

Comparação (classes de complexidade)	**H, *p***	**Conclusão** ($\alpha = 0.05$)
Tarefas Textuais	6.65, 0.036	**Existe uma relação entre as mudanças de complexidade e as Repostas.**
Tarefas Gráficas	0.4, 0.8187	Não existe relação entre mudanças de complexidade e as Repostas

Tabela 7.16: Resumo análise *Kruskall-Wallis post hoc* com *Mann-Whitney* para cenários textuais

Comparação (classes de complexidade)	**U, *p***	**Conclusão** ($\alpha = 0.05$)
Baixa vs Média	105.0, 0.76	Não existe diferença significativa.
Média vs Alta	163.5, 0.03	**Existe diferença significativa**.
Baixa vs Alta	168.0, 0.02	**Existe diferença significativa**.

Na análise de variância (*Mann-Whitney*), fixando a complexidade do cenário e variando o tipo de cenário, os resultados mostraram que não existiram diferenças significativas na complexidade *Média* e *Baixa*. Já na classe de complexidade *Alta* registou-se diferenças significativas para afirmar que o tipo de cenário influenciou as repostas registadas. Os resultados da análise são apresentados na Tabela 7.17. Todos os resultados apresentados foram obtidos um grau de confiança de 95%.

Tabela 7.17: Resumo análise *Mann-Whitney* - Fixando a complexidade e variando o tipo de cenário

Comparação (tipo de cenário)	**U, *p***	**Conclusão** ($\alpha = 0.05$)
Baixa	112.5, 1.00	Não existe relação entre as Respostas e o tipo de cenário.
Média	111.5, 0.97	Não existe relação entre as Respostas e o tipo de cenário.
Alta	172.5, 0.01	**Existe relação entre as Respostas e o tipo de cenário**.

Atenção, Meditação e HRV

Em relação aos sinais capturados através dos sensores biométricos, os resultados não foram satisfatórios. Não foi possível ter uma captura positiva de muitos dos dados dos participantes. Relativamente às tarefas *TC1* e *TC2* foram capturados 4 registos. Para a tarefa *TC3* e *TC4* foram capturados 7 registos e para as tarefas *TC5* e *TC6* 6 registos.

Dos registos capturados, foi calculado a média de atenção e meditação para cada par de tarefa realizada. As Figuras 7.24 e 7.25 mostram os resultados. O tratamento dos dados efetuado foi remover os cinco segundos iniciais e finais da captura da informação, que correspondiam ao tempo de iniciar a captura do sensor e mudar para a vista da tarefa e vice-versa.

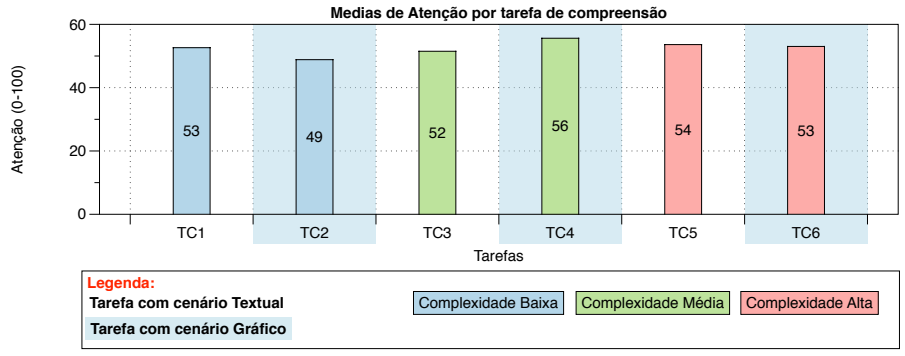


Figura 7.24: Média dos níveis de Atenção das tarefas de compreensão

Em relação aos batimentos cardíacos, registaram-se leituras repentinas de 150 a 200 batimentos cardíacos por minuto, constantes. Essas leituras são consideradas erradas porque batimentos dessa grandeza estão relacionados com pessoas em extremo esforço físico. Os participantes do experimento não sofriam problemas cardíacos, nem apresentavam sintomas de enorme cansaço físico. A Figura 7.26 mostra um exemplo do caso mencionado.

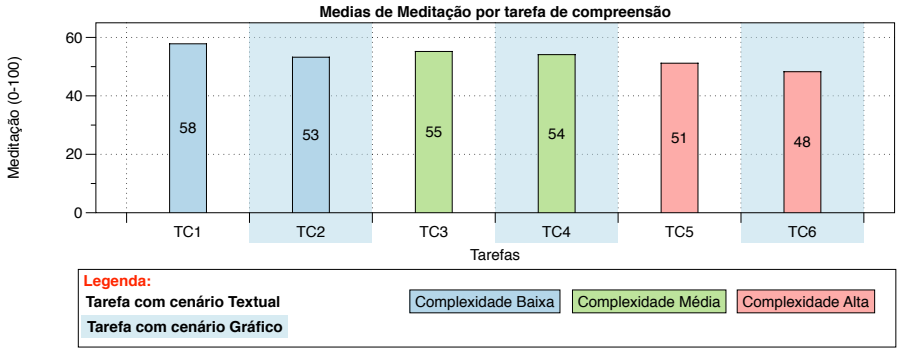


Figura 7.25: Média dos níveis de Meditação das tarefas de compreensão

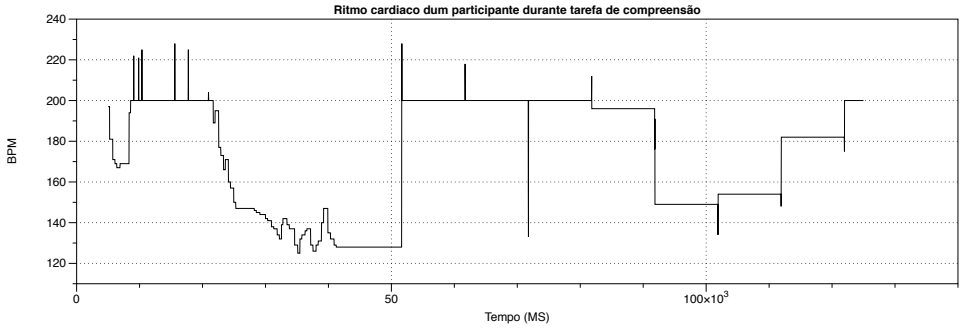


Figura 7.26: Exemplo de Leitura dos Batimentos Cardiacos

Os resultados conseguidos não são suficientes para conseguir concluir algo com a realização do experimento, portanto foram descartados da avaliação. Não foi repetido o experimento devido à escassez de tempo e o facto de não serem conhecidos os motivos de erro na captura dos sinais.

7.4.3 Resumo dos dados obtidos

A Figura 7.27 mostra as médias de todas as tarefas de compreensão. A variância da média nos cenários textuais é muito acentuada para a complexidade *Alta*, em comparação com os cenários gráficos, como mostram os resultados da análise da variância. De notar também que os resultados capturados com o questionário final, que avaliaram a dificuldade das tarefas de acordo com o ponto de vista dos participantes, é semelhante à capturada pelos questionários do NASA-TLX. Foi feita a correlação de *Spearman* entre as dificuldades obtidas nos questionários e os *workloads* capturados pelo NASA-TLX para todas as tarefas e o resultado ($r[118] = 0.69602$, $p = 0$) mostrou uma correlação positiva e relevante. Também foi aplicada a correlação de *Spearman* entre o *tempo* e o *workload* (NASA-TLX) e o resultado ($r[118] = 0.22949$, $p = 0.01169$) mostrou uma correlação positiva entre os dois parâmetros, mas não relevante.

Através deste experimento podemos afirmar que o *BehaviorMap* mostrou melhor consistência de resultados em relação aos cenários textuais. O resumo dos resultados está

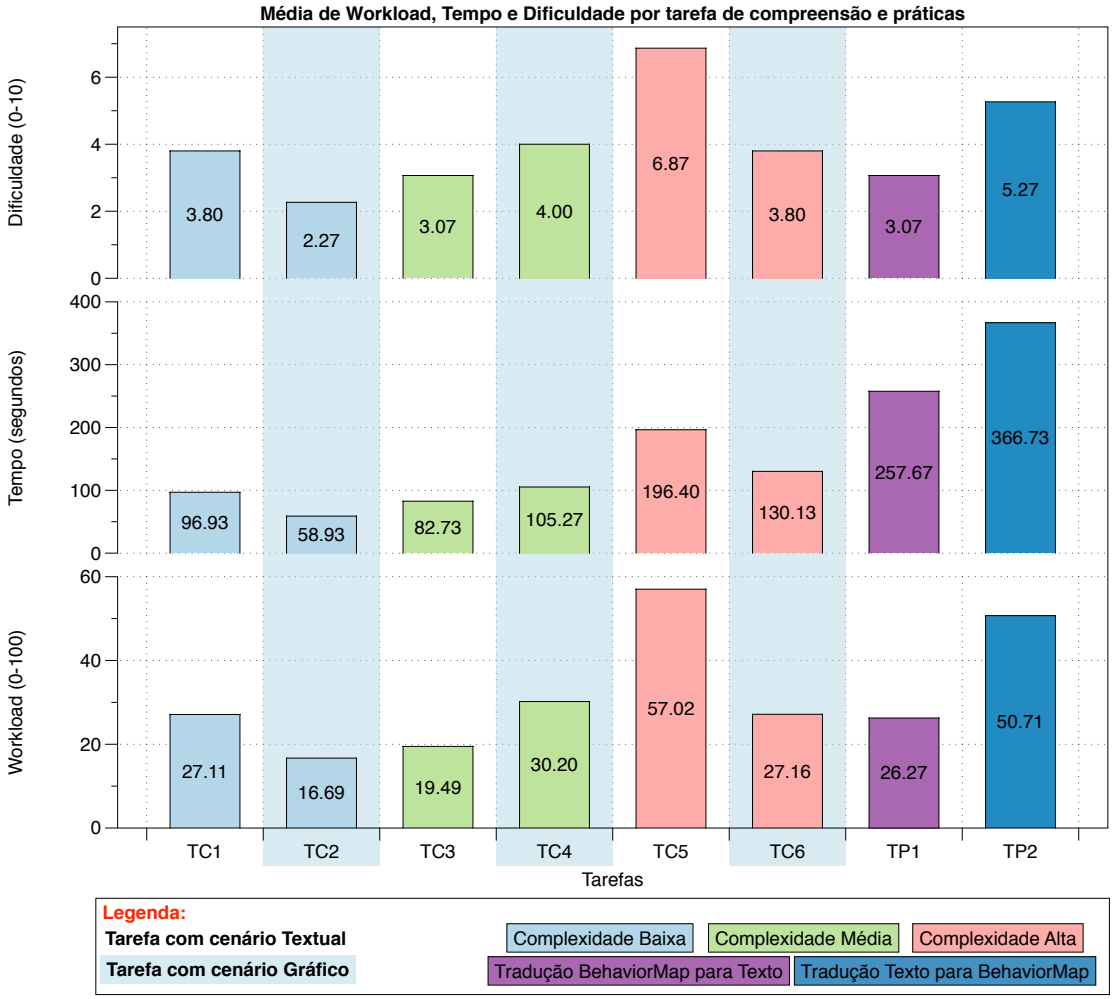


Figura 7.27: Médias de Workload (NASA-TLX), Tempo de Execução e Dificuldade das tarefas

apresentado nas Tabelas 7.18, 7.19 e 7.20. Nas tarefas práticas os resultados foram claramente mais favoráveis para a tradução de *BehaviorMap* para texto. As dificuldades dos participantes foram anotadas durante a execução da tarefa e questionadas após a realização do experimento. Os participantes responderam que tiveram dificuldade em conseguir distinguir o que era entidade, atributo e valor na tradução do texto para o mapa mental. Mesmo quem obteve melhores resultados na tradução de texto para mapa mental levou mais tempo a consegui-lo. De notar que o uso da ferramenta *BehaviorMap* pode ter influenciado a tradução do texto para o mapa mental em alguns casos, mas a razão principal apontada pelos participantes foi a incapacidade de conseguir distinguir os elementos no texto.

Tabela 7.18: Resumo dos resultados obtidos das tarefas práticas

Medição	Resultado
Workload	Com diferenças significativas (**melhor resultado na tradução de BehaviorMap para texto**)
Esforço Temporal	Com diferenças significativas (**melhor resultado na tradução de BehaviorMap para texto**)
Respostas (% Certas)	Com diferenças significativas (**melhor resultado na tradução de BehaviorMap para texto**)

Recaindo na análise das tarefas de compreensão, fixando o tipo de cenário e variando as classes de complexidade, os cenários gráficos mantiveram um desempenho constante, onde apenas existiu uma diferença significativa no esforço temporal da classe baixa para as restantes, que era bastante inferior. No entanto, isso é vantajoso porque demonstra que os participantes conseguiram perceber os cenários mais rapidamente. Já nos cenários textuais, existiram sempre diferenças significativas nas três medições para o cenário de complexidade *Alta*. Na análise onde se fixou a classe de complexidade e variou o tipo de cenário, registou-se sempre diferenças significativas entre os cenários de complexidade *Alta*, onde os cenários gráficos tiveram melhor desempenho. Também foi registado um melhor desempenho no esforço temporal no cenário de complexidade *Baixa*, onde o cenário gráfico teve melhor desempenho.

Tabela 7.19: Resumo dos resultados obtidos para tarefas compreensão - Comparação fixando o tipo de cenário e variando a complexidade

Medição	Cenário	Resultado
Workload	Textual	Diferenças significativas na complexidade *Alta* (Superior ao restante)
	Gráfico	**Não foram registadas diferenças significativas**
Esforço Temporal	Textual	Diferenças significativas na complexidade *Alta* (Superior ao restante)
	Gráfico	**Diferenças significativas na complexidade *Baixa* (Inferior ao restante)**
Respostas (% Certas)	Textual	Diferenças significativas na complexidade *Alta* (Superior ao restante)
	Gráfico	**Não foram registadas diferenças significativas**

Os comentários dos participantes do experimento são apresentados na Tabela 7.21.

Tabela 7.20: Resumo dos resultados obtidos para tarefas compreensão - Comparação fixando a classe de complexidade e variando o tipo de cenário

Medição	Complexidade	Resultado
Workload	Baixa	Sem diferenças significativas
	Média	Sem diferenças significativas
	Alta	**Com diferenças significativas (melhor resultado para *BehaviorMap*)**
Esforço Temporal	Baixa	**Com diferenças significativas (melhor resultado para *BehaviorMap*)**
	Média	Sem diferenças significativas
	Alta	**Com diferenças significativas (melhor resultado para *BehaviorMap*)**
Respostas (% Certas)	Baixa	Sem diferenças significativas
	Média	Sem diferenças significativas
	Alta	**Com diferenças significativas (melhor resultado para *BehaviorMap*)**

Tabela 7.21: Comentários dos participantes sobre o experimento

#	Comentario
1	Pode haver uma certa ambiguidade ou má interpretação na parte escrita. Na parte gráfica, há sempre um padrão bastante definido a seguir, que é de fácil interpretação e não requer muito esforço por parte do leitor. No exemplo TP2, eu interpretei o alert status como uma entidade, mas era um atributo da stock. Ou seja, interpretei mal o cenário. Para a escrita ser bem sucedida, deve ser redundante e conseguirmos destacar muito bem as partes que levam ao gráfico do cenário.
2	Em relação ao enjoyment, gostava de haver duas opções, pois a experiência foi parte interessante (modelos visuais) e parte fadante e frustante (dos textos). A ferramente é bem intuitiva, depois de bem apresentado os seus prinícipios básicos, bem como de fácil de usar. Não sou da área computacional, por isso, a parte dos textos e todas as especificações me pareciam mais dificeis de entender.
3	Ferramenta bastante intuitiva e perceptível. Os exercícios práticos, tendo em conta que se tem a template no TP1 torna-se bastante fácil e no TP2 a ferramenta simples e intuitiva ajuda muito no desenvolvimento do que se pretende. Os exercícios de compreensão são também simples.
4	Os exercícios tornaram-se mais fáceis com a prática. Achei o mapa mental mais rápido e simples de interpretar do que o cenário textual.
5	Representação em diagrama mais perceptível. Ferramentas simples, intuitivas e de fácil compreensão.
6	I will mostly critique the forms used to specify behaviours: graphical and textual. As for the textual representation, it has the advantage of being very similar to the English language, allowing an easy specification of the behaviour. This, however, is also a negative point. As the language has a specific grammar, it may become difficult for the user to specify the behaviour that way, because the way he would phrase it in English may not necessarily correspond to the textual language's grammar. The graphical representation is very useful, as each node is very specific about its purpose, and connections are easier to understand. Also, I would like to add that, after doing 4 of the tests, I felt a slight headache.
7	If the text isn't too complex and/or contains too much information it's more simple to read the diagram. I find a small concise text to be more effective than a diagram, yet a diagram is better than a longer and more complex text.
8	The tool is easy to use and pretty straight foward. The exercises, in general, are easy although one or two require a little more brain power.
9	Gostei mais de TP2 por se tratar do processo simples e demonstrativo de um limite enquanto que em TP1, é simples e funcional também, mas contém dois objectos que se dependem um do outro e por isso foi mais divertido. Prefiro as condições vistas em modo de diagrama como no TP1 pela exemplificação e indicação do atributo em linha e por seta.
10	De modo geral, acho os esquemas mais fáceis de perceber que o texto porque é mais fácil perceber os componentes de cada parte. Não sei se devido ao tema ou à quantidade de informação, mas tive mais facilidade com os esquemas que continham mais informação. Relativamente aos textos, alguns achei difícil identificar os componentes e depois ficou difícil transpô-los para esquemas.
11	Há situações em que a narrativa me ajuda a ter uma melhor percepção do que se pretende em relação ao diagrama (por ex. TC5); nas outras situações, grosso modo, achei os diagramas deveras explícitos.
12	Achei textual mais fácil, embora alguns diagramas eram facilmente compreensíveis.

7.5 Discussão

Linguagens

Relativamente ao *DomainMap*, a sua construção foi baseada em tentar capturar apenas os elementos principais de um modelo de domínio, baseado na análise feita por *Eric Evans* [Eva03]. Não foi o foco principal da dissertação, mas foi abraçada como uma oportunidade de tentar criar um *framework* mais abrangente com a premissa de que os mapas mentais garantiam a proximidade dos *stakeholders* no processo de desenvolvimento graças às características cognitivas presentes. Essa hipótese não foi testada nas avaliações conduzidas devido a escassez de tempo. É um ponto de trabalho futuro a realizar para verificar se a aposta nos mapas mentais é promissora.

A capacidade de inserir regras de negócio (sob a forma de regras OCL) é outro ponto grande de melhoria no *DomainMap*, pois pode ser uma forma de conseguir inserir regras de negócio no início de desenvolvimento, aprovadas por especialistas de domínio e garantidas em artefactos futuros.

Outro aspecto importante é tentar averiguar se o *DomainMap* deve seguir uma linha de tentar "substituir" o UML como linguagem de modelação de modelos conceptuais ou auxiliar na sua construção. É necessária esta decisão para descobrir qual o esforço que deve ser feito na evolução da linguagem. No entanto, o principal objetivo é perceber se de facto existe facilidade em ler mapas mentais em relação a outras linguagens de modelação.

Recaindo sobre o *BehaviorMap*, esta linguagem foi o foco da dissertação e teve mais escrutínio na sua construção. Baseamos a sua estrutura na *Lógica de Hoare*, como muitas outras linguagens de verificação formal o fizeram. No entanto, não fizemos nenhuma comparação formal com os seus conceitos. A DSL implementada aposta na facilidade da escrita de cenários com a incorporação de modelos de domínio do *DomainMap* para dar continuidade à informação capturada sob o mesmo modelo. Essa incorporação traz vantagens como a validação de informação escrita nos cenários com mecanismos de sugestão de nomes e notações visuais validadas pelos próprios *stakeholders*. O *BehaviorMap* apresenta uma outra vantagem para a abordagem textual que é a sua construção de testes de aceitação, automática e com um maior número de verificações sobre o cenário especificado. A transformação é feita utilizando técnicas de transformação de modelo-para-texto para minorar possíveis erros de transformações manuais. Ainda assim o *BehaviorMap* não conseguiu solucionar alguns problemas do BDD, tais como:

1. Capacidade de representar aspectos de *interfaces*;
2. Capacidade de representar cenários com múltiplos passos *When Then*;
3. Capacidade de condições temporais (ex. quando passar 3 segs);
4. Capacidade de especificar requisitos *cross-cutting*.

Além desses problemas, o *BehaviorMap* tem pontos de melhoria, nomeadamente nas

suas transformações, a capacidade de melhorar a transformação para minorar refinamentos e a especificação da ordenação de execução de múltiplas ações.

Aplicabilidade

A avaliação da aplicabilidade dos cenários mostrou que o *BehaviorMap* consegue especificar muitos cenários capturados de diversas fontes, excetuando alguns casos específicos. Um ponto importante dos cenários recolhidos é a sua origem. Muitos deles são provenientes de manuais de BDD e não são exemplos de aplicações reais. No entanto, não foram encontradas muitas fontes com trabalhos relacionados de BDD. As traduções dos cenários textuais para *BehaviorMap* foram baseadas num conjunto de regras, mas em caso de alguma subjectividade no texto, as decisões de transformar os elementos textuais em elementos do *BehaviorMap* foram tomadas pelo autor da dissertação. Para que essas decisões fossem completamente corretas e validadas, seria necessário possuir mais informações do domínio do cenário. Isso não aconteceu porque os cenários eram apresentados isoladamente sem mais informação. Por último, mostramos também que a abordagem do *BehaviorMap* contempla mais casos de testes em relação à abordagem textual.

Compreensão

Este primeiro experimento inicial mostrou algumas evidências que os cenários escritos com mapas mentais são mais fáceis de entender relativamente aos cenários textuais. Apesar disso, é necessário a produção de mais experimentos com mais pessoas e com diferentes cenários. O experimento partiu de métricas de avaliação de cenários BDD propostas pelos autores da dissertação, métricas essas que podem não estar devidamente validadas e necessitarem de modificações. Apesar disso, os cenários escolhidos para este experimento conseguiram abranger diferentes tipos de cenários, mostrando que os cenários textuais são mais complicados de compreender em relação aos gráficos.

Nomeadamente, nas tarefas práticas, os resultados mostraram uma clara facilidade em traduzir corretamente um cenário gráfico para texto em relação a uma tradução textual para gráfico. A análise realizada admitiu que os participantes sabiam utilizar a DSL de suporte do *BehaviorMap* e tiveram dificuldade apenas na tradução. Esta assunção não esta completamente correta, apesar do treino oferecido no início do experimento. No entanto, os participantes quando confrontados em relação à tarefa, mencionaram o facto de não conseguirem distinguir os elementos no cenário textual para realizar a tradução.

Nas tarefas de compreensão, as dificuldades encontradas ao utilizar o *MindWave* e *PulseSensor* deveram-se à escassez de tempo para testar devidamente o equipamento antes da realização do experimento. Os aparelhos não tinham sido testados anteriormente a este experimento. Foi feito um esforço inicial para criar uma pequena plataforma para registar as ondas cerebrais e os batimentos cardíacos e foram realizados testes com o autor da dissertação, mas na altura do experimento ocorrerem falhas de comunicação com os aparelhos e as aplicações, que levaram ao insucesso de captura da informação. Durante o

experimento tentou-se minorar os erros e perceber o porquê das falhas, mas sem sucesso. Não foi utilizada a plataforma de análise oficial do equipamento devido à falta de fundos monetários para a sua compra.

Relativamente aos resultados, o *BehaviorMap* teve melhores resultados com o aumento de complexidades dos cenários, no entanto, os cenários textuais tiveram bons desempenhos nas classes de complexidade *Baixa* e *Média*. Mesmo sentindo dificuldade nos cenários de complexidade *Alta*, obteve-se uma média de 60% de respostas certas nos cenários. Neste experimento inicial existe uma pequena evidência que, em certos casos, a informação apresentada graficamente tem vantagens em relação à informação apresentada textualmente, mas seria necessário continuar o experimento para conseguir evidenciar os problemas dos cenários gráficos e textuais.

Ameaças à validade

A avaliação desenhada neste experimento propunha analisar a dificuldade cognitiva de cenários comportamentais BDD textuais e gráficos. Foi proposto utilizar meios de avaliação subjectivos como perguntas sobre o modelo, esforço temporal e o questionário NASA-TLX. Para combater a subjectividade dessas medições foram utilizados aparelhos de medição de sinais biométricos, de forma a conseguir interpretar o esforço cognitivo de cada participante, ao realizar as tarefas. A captura dos sinais biométricos não correu como esperado levando a uma exclusão completa dos seus resultados na avaliação. Assim, sobrou a parte subjectiva da avaliação. Apesar disso, surgiu um padrão semelhante nos vários meios de avaliação de avaliação (NASA-TLX, esforço temporal, respostas, dificuldade), levando aos autores a pensar que os resultados da avaliação são consistentes entre si em relação ao esforço cognitivo dos participantes.

Outro aspecto que pode comprometer esta avaliação é a dimensão de participantes do experimento. Foram utilizados 15 indivíduos para o experimento, sem nenhum critério de seleção em relação a uma especialização. O total de participantes é reduzido para conseguir afirmar que um modelo é mais facilmente compreensível que outro, porém, devido às restrições temporais da dissertação de mestrado, não foi possível continuar com o experimento. Este resultado promove é a realização de futuros experimentos para verificar se existe uma tendência nesta conclusão inicial.

7.6 Resumo

Neste capítulo foram abordadas as avaliações efetuadas nesta dissertação. A primeira avaliação diz respeito à aplicabilidade do *BehaviorMap* em conseguir especificar diversos cenários textuais BDD. Foram recolhidos múltiplos cenários textuais e verificado se o *BehaviorMap* conseguia especificar o seu comportamento. Além disso, foi também realizada uma comparação da cobertura de asserções feitas entre o *BehaviorMap* e as abordagens textuais. O *BehaviorMap* conseguiu superar as asserções em relação às abordagens

textuais.

A segunda avaliação apresentada diz respeito à compreensão dos cenários. Tentou-se replicar um experimento com o uso de sensores biométricos para avaliar o esforço cognitivo ao compreender cenários textuais e gráficos. A utilização dos sensores não correu como esperado e os seus dados não conseguiram ser utilizados. No entanto, através de outros parâmetros de avaliação (e.g. questionário, tempo e perguntas) consegue-se concluir que os cenários gráficos tiveram um desempenho superior aos cenários textuais.

8

Conclusão

Esta dissertação propôs uma abordagem ao problema da comunicação entre *stakeholders* e a equipa de desenvolvimento e no melhoramento da prática BDD. A proposta fundou-se na construção de duas linguagens gráficas para representar comportamentos BDD e modelos de domínio, ambas baseadas em modelos cognitivos, nomeadamente mapas mentais.

8.1 Contribuições

A primeira contribuição desta dissertação é a construção de modelos com base em modelos cognitivos, nomeadamente mapas mentais. Esta tentativa de criar modelos centrados no utilizador serve para investigar um novo método de tentar integrar *stakeholders* no processo de desenvolvimento. Foram construídos dois modelos para modelar modelos de domínio e cenários comportamentais BDD.

A segunda contribuição é a linguagem desenvolvida, *BehaviorMap*. O *BehaviorMap* especifica modelos comportamentais sob a sintaxe do BDD e pretende melhorar, não só aspectos cognitivos como também questões relacionadas com a verificação na escrita do cenário. A linguagem melhora também aspectos relacionados com a escrita dos testes, através da geração automática dos mesmos, garantindo melhor cobertura em comparação com outras aplicações existentes. Em relação à verificação e auxílio na escrita dos cenários, a linguagem tem restrições próprias do BDD e utiliza informação do domínio do problema através da linguagem *DomainMap*.

A avaliação sobre a compreensão de cenários BDD por parte de utilizadores comuns mostrou uma pequena evidência que os cenários *BehaviorMap* são mais facilmente compreendidos do que cenários textuais. A partir deste resultado conseguimos mostrar que

existe a possibilidade de diminuir a dificuldade cognitiva para entender um modelo de requisitos por parte dos *stakeholders*. No entanto, não foi demonstrado se isso tem um efeito no aumento da participação dos *stakeholders* no desenvolvimento de *software*. De forma a compreender melhor essa situação, mais experimentos tinham de ser realizados, verificando diversos aspectos.

A terceira contribuição é a linguagem *DomainMap*. Foi a segunda linguagem desenvolvida e o seu propósito é capturar informação do domínio de aplicação. A linguagem utiliza também um modelo baseado em mapas mentais, sendo esta a premissa desta dissertação. Esta linguagem comunica com o *BehaviorMap* de forma que a informação validada seja propagada ao longo do desenvolvimento. Consegue também transformar os modelos desenvolvidos em classes *Java*, criando artefactos de programação com entidades validadas por *stakeholders*.

Em suma, esta dissertação desenvolveu duas linguagens para tentar combater a dificuldade de compreensão de modelos de requisitos por parte de *stakeholders* e para melhorar certos aspectos do BDD. As linguagens foram inseridas no processo de desenvolvimento do BDD e funcionam em conjunto. A linguagem *BehaviorMap* introduziu melhorias como validação sintática, impondo regras na escrita, e validação de conteúdo através da importação da informação capturada pela linguagem *DomainMap*. Melhorou também o processo de transformação para casos de testes, utilizando transformações automáticas, e garantindo uma maior cobertura de mais casos de testes em relação às ferramentas atuais. De forma a facilitar a satisfação dos casos de testes criados, a linguagem *DomainMap* tem a funcionalidade de transformar a informação do domínio em código *Java*.

8.2 Trabalho futuro

Com a realização desta dissertação, existem várias possibilidades de trabalho futuro. Existe a necessidade de continuar os experimentos para concluir se tantos os modelos do *DomainMap* e do *BehaviorMap* são mais facilmente compreendidos por *stakeholders* em relação aos seus rivais e se essa facilidade de compreensão compensa no desenvolvimento de *software*. Além de uma avaliação cognitiva, uma avaliação de usabilidade das DSLs também é necessária para verificar se existem aspectos de funcionalidades que necessitam ser remodelados. Sem esquecer também que os processos de desenvolvimento não foram validados em nenhum projecto real e podem necessitar de refinamentos.

No domínio do *DomainMap*, a evolução da linguagem é um ponto de discussão. Se deve-se focar em auxiliar na construção de UML ou tentar substituí-lo e a sua capacidade de inserção de regras de negócio. Tentar descobrir uma forma de inserir regras de negócio que consigam ser compreendidas e validadas por *stakeholders* e simultaneamente mantidas noutros artefactos de desenvolvimento. Abordar também questões de escalabilidade do *DomainMap*, verificando qual a capacidade máxima de informação que é capaz de apresentar mantendo a usabilidade máxima para os seus leitores.

Relativamente ao *BehaviorMap*, um ponto de melhoria seria na construção de testes

de aceitação, minorando ao máximo refinamentos necessários. Também a capacidade de conseguir ordenar a execução de múltiplas acções para uma melhor definição de cenários. Caso a imposição de regras de negócio seja imposta no *DomainMap*, ter um mecanismo de validação dessas restrições aplicados no *BahaviorMap*, inserido assim mais mecanismos de validação na especificação de comportamento.

Sendo mais ambicioso, caso as avaliações sobre a compreensão dos modelos melhore a colaboração dos *stakeholders* no desenvolvimento de *software*, uma aposta a considerar seria a transposição do *DomainMap* e do *BehaviorMap* para aplicações *Web*. Isto permitiria o uso mais facilitado por todos os membros do desenvolvimento.

Bibliografia

[Abd+11] N. N. B. Abdullah, S. Honiden, H. Sharp, B. Nuseibeh e D. Notkin. "Communication patterns of agile requirements engineering". Em: *Proceedings of the 1st Workshop on Agile Requirements Engineering*. AREW '11. Lancaster, United Kingdom: ACM, 2011, 1:1–1:4. ISBN: 978-1-4503-0890-8. DOI: 10.1145/2068783.2068784. URL: http://doi.acm.org/10.1145/2068783.2068784.

[Amb14] S. Ambler. *Agile Modeling (AM) - Effective Practices for Modeling and Documentation*. Acedido em Janeiro de 2014. URL: http://www.agilemodeling.com.

[Amb02] S. W. Ambler. *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*. Boston: Wiley, mar. de 2002. ISBN: 0471202827. URL: http://www.amazon.fr/exec/obidos/ASIN/0471202827/citeulike04-21.

[AC14] S. Ambler e A. Cockburn. *ModesOfCommunication*. Acedido em Janeiro de 2014. URL: http://www.agilemodeling.com/essays/communication.htm.

[AD52] T. W. Anderson e D. A. Darling. "Asymptotic Theory of Certain 'Goodness of Fit' Criteria Based on Stochastic Processes". Em: *The Annals of Mathematical Statistics* 23.2 (jun. de 1952), pp. 193–212. URL: http://dx.doi.org/10.1214/aoms/1177729437.

[And07] J. Andrea. "Envisioning the Next-Generation of Functional Testing Tools". Em: *Software, IEEE* 24.3 (mai. de 2007), pp. 58–66. ISSN: 0740-7459. DOI: 10.1109/MS.2007.73.

[And00] J. L. Andreassi. "Psychophysiology: Human Behavior Physiological Response". Em: (2000).

[And14] Android. *Iconography*. Acedido em Agosto de 2014. URL: https://developer.android.com/design/style/iconography.html.

[App14] Apple. *Icon Design Guidelines*. Acedido em Agosto de 2014. URL: https://developer.apple.com/library/mac/documentation/UserExperience/Conceptual/AppleHIGuidelines/IconsImages/IconsImages.html.

[Ara+05] J. Araújo, E. Baniassad, P. Clements, A. Moreira, A. Rashid e B. Tekinerdogan. *Early Aspects: The Current Landscape*. Rel. téc. Lancaster University, 2005.

[Bak+13] K. Bak, D. Zayan, K. Czarnecki, M. Antkiewicz, Z. Diskin, A. Wasowski e D. Rayside. "Example-driven modeling: model = abstractions + examples". Em: *ICSE*. Ed. por D. Notkin, B. H. C. Cheng e K. Pohl. IEEE / ACM, 2013, pp. 1273–1276. ISBN: 978-1-4673-3076-3.

[Ban07] T. J. Bang. "An Agile Approach to Requirement Specification". Em: *Proceedings of the 8th International Conference on Agile Processes in Software Engineering and Extreme Programming*. XP'07. Como, Italy: Springer-Verlag, 2007, pp. 193–197. ISBN: 978-3-540-73100-9. URL: http://dl.acm.org/citation.cfm?id=1768961.1769005.

[BAG12] A. Barisic, V. Amaral e M. Goulão. "Usability Evaluation of Domain-Specific Languages". Em: *QUATIC*. 2012, pp. 342–347.

[Bar+10] J. B. Barlow, J. S. Giboney, M. J. Keith, D. W. Wilson, R. M. Schuetzler, P. B. Lowry e A. Vance. "Overview and Guidance on Agile Development in Large Organizations". Em: *Communications of the Association for Information Systems* 29.2 (2010). Available at: http://aisel.aisnet.org/cais/vol29/iss1/2.

[BE] Z. A. Barmi e A. H. Ebrahimi. *Automated testing of non-functional requirements based on behavioural scripts*.

[Bec02] Beck. *Test Driven Development: By Example*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN: 0321146530.

[BA04] K. Beck e C. Andres. *Extreme Programming Explained: Embrace Change (2nd Edition)*. Addison-Wesley Professional, 2004. ISBN: 0321278658.

[Bec+14] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland e D. Thomas. *Manifesto for Agile Software Development*. Acedido em Janeiro de 2014. URL: http://www.agilemanifesto.org/.

[Beg+14] A. Begel, T. Fritz, S. Mueller, S. Yigit-Elliott e M. Zueger. "Using Psycho-Physiological Measures to Assess Task Difficulty in Software Development". Em: *Proceedings of the International Conference on Software Engineering*. International Conference on Software Engineering, jun. de 2014. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=209878.

[Ber+00] C. Berka, D. J. Levendowski, M. N. Lumicao, A. Yau, G. Davis, V. T. Zivkovic, R. E. Olmstead, P. D. Tremoulet e P. L. Craven. "EEG Correlates of Task Engagement and Mental Workload in Vigilance, Learning, and Memory Tasks". Em: *Aviation, Space, and Environmental Medicine* 78.5 (2007-05-01T00:00:00), B231–B244. URL: http://www.ingentaconnect.com/content/asma/asem/2007/00000078/A00105s1/art00032.

[Bød04] K. Bødker. *Participatory IT design : designing for business and workplace realities*. Cambridge, Mass: MIT Press, 2004. ISBN: 0262512440.

[Boo94] G. Booch. *Object-oriented Analysis and Design with Applications (2Nd Ed.)* Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc., 1994. ISBN: 0-8053-5340-2.

[Bro09] D. Brolund. "Documentation by Example". Em: *Agile Processes in Software Engineering and Extreme Programming*. Ed. por P. Abrahamsson, M. Marchesi e F. Maurer. Vol. 31. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2009, pp. 251–252. ISBN: 978-3-642-01852-7. DOI: 10.1007/978-3-642-01853-4_54. URL: http://dx.doi.org/10.1007/978-3-642-01853-4_54.

[BW93] K. A. Brookhuis e D. de Waard. "The use of psychophysiology to assess driver status". Em: *Ergonomics* 36.9 (1993), pp. 1099–1110.

[BWS96] J. B. Brookings, G. F. Wilson e C. R. Swain. "Psychophysiological responses to changes in workload during simulated air traffic control". Em: *Biological Psychology* 42.3 (1996). Psychophysiology of Workload, pp. 361–377. ISSN: 0301-0511. DOI: http://dx.doi.org/10.1016/0301-0511(95)05167-8. URL: http://www.sciencedirect.com/science/article/pii/0301051195051678.

[Bro87] F. P. Brooks Jr. "No Silver Bullet Essence and Accidents of Software Engineering". Em: *Computer* 20.4 (abr. de 1987), pp. 10–19. ISSN: 0018-9162. DOI: 10.1109/MC.1987.1663532. URL: http://dx.doi.org/10.1109/MC.1987.1663532.

[Buz14] T. Buzan. *Mind Map Example*. Acedido em Janeiro de 2014. URL: http://www.tonybuzan.com/images/mm_health.jpg.

[BB93] T. Buzan e B. Buzan. *The Mind Map Book*. 1ª ed. London: BBC Books, 1993.

[Cai+13] P. Caire, N. Genon, P. Heymans e D. L. Moody. "Visual notation design 2.0: Towards user comprehensible requirements engineering notations". Em: *RE*. 2013, pp. 115–124.

[CR08] L. Cao e B. Ramesh. "Agile Requirements Engineering Practices: An Empirical Study". Em: *IEEE Software* 25.1 (2008), pp. 60–67. ISSN: 0740-7459. DOI: http://doi.ieeecomputersociety.org/10.1109/MS.2008.1.

[CIG13] Á. Carrera, C. Iglesias e M. Garijo. "Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development". English. Em: *Information Systems Frontiers* (2013), pp. 1–14. ISSN: 1387-3326. DOI: 10.1007/s10796-013-9438-5. URL: http://dx.doi.org/10.1007/s10796-013-9438-5.

[CCM10] R. A. de Carvalho, F. L. de Carvalho e Silva e R. S. Manhães. "Mapping Business Process Modeling constructs to Behavior Driven Development Ubiquitous Language". Em: *CoRR* abs/1006.4892 (2010).

[Car+13] R. Carvalho, F. Carvalho e Silva, R. Manhães e G. Oliveira. "Implementing Behavior Driven Development in an Open Source ERP". Em: *Enterprise Information Systems of the Future*. Ed. por G. Poels. Vol. 139. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, 2013, pp. 242–249. ISBN: 978-3-642-36610-9. DOI: 10.1007/978-3-642-36611-6_22. URL: http://dx.doi.org/10.1007/978-3-642-36611-6_22.

[Che+10] D. Chelimsky, D. Astels, B. Helmkamp, Z. Dennis, D. North e A. Hellesoy. *The Rspec Book: Behaviour Driven Development With Rspec, Cucumber, and Friends*. Pragmatic Bookshelf Series. Pragmatic Programmers, LLC, 2010. ISBN: 9781934356371. URL: http://books.google.pt/books?id=0rxoPgAACAAJ.

[Che12] S. Cheng. "Behaviour Driven Development and integration platform". Tese de mestrado. Hong Kong Polytechnic University, 2012. URL: http://theses.lib.polyu.edu.hk/handle/200/6624.

[Cin+13] B. Cinaz, B. Arnrich, R. La Marca e G. Tröster. "Monitoring of mental workload levels during an everyday life office-work scenario". English. Em: *Personal and Ubiquitous Computing* 17.2 (2013), pp. 229–239. ISSN: 1617-4909. DOI: 10.1007/s00779-011-0466-1. URL: http://dx.doi.org/10.1007/s00779-011-0466-1.

[Coc06] A. Cockburn. *Agile software development: the cooperative game (agile software development series)*. Addison-Wesley Professional, 2006. URL: http://dl.acm.org/citation.cfm?id=1177327.

[Coc02] A. Cockburn. *Agile software development*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN: 0-201-69969-9.

[CB10] J. O. Coplien e G. Bjrnvig. *Lean Architecture: For Agile Software Development*. Wiley Publishing, 2010. ISBN: 0470684208, 9780470684207.

[CLG12] J. S. Cuadrado, J. de Lara e E. Guerra. "Bottom-Up Meta-Modelling: An Interactive Approach". Em: *MoDELS*. Ed. por R. B. France, J. Kazmeier, R. Breu e C. Atkinson. Vol. 7590. Lecture Notes in Computer Science. Springer, 2012, pp. 3–19. ISBN: 978-3-642-33665-2.

[Dan13] C. S. Dan Elbaum. "The Perfect Couple: Domain Models Behavior-Driven Development". Em: 2013. URL: http://www.uploads.pnsqc.org/2013/papers/t-100_Elbaum_paper.pdf.

[Dan+13] M. Daneva, E. Van Der Veen, C. Amrit, S. Ghaisas, K. Sikkel, R. Kumar, N. Ajmeri, U. Ramteerthkar e R. Wieringa. "Agile Requirements Prioritization in Large-scale Outsourced System Projects: An Empirical Study". Em: *J. Syst. Softw.* 86.5 (mai. de 2013), pp. 1333–1353. ISSN: 0164-1212. DOI: 10.1016/j.jss.2012.12.046. URL: http://dx.doi.org/10.1016/j.jss.2012.12.046.

[De 90] M. H. De Jong PJ. "Eyeblink frequency, rehearsal activity, and sympathetic arousal". Em: *Int'l. Journal of Neuroscience* 51(1-2):89–94 (1990).

[Die+12] M. Diepenbeck, M. Soeken, D. Grose e R. Drechsler. "Behavior Driven Development for circuit design and verification". Em: *High Level Design Validation and Test Workshop (HLDVT), 2012 IEEE International*. Nov. de 2012, pp. 9–16. DOI: 10.1109/HLDVT.2012.6418237.

[Die+13] M. Diepenbeck, M. Soeken, D. Große e R. Drechsler. "Towards Automatic Scenario Generation from Coverage Information". Em: *8th International Workshop on Automation of Software Test*. IEEE, 2013.

[Doe57] D. G. Doehring. "THE RELATION BETWEEN MANIFEST ANXIETY AND RATE OF EYEBLINK IN A STRESS SITUATION". Em: *Central Institute for the Deaf, St Louis* (1957).

[Don12] C. Donnat. "Behaviour driven development (BDD)". Tese de mestrado. Haute école de gestion de Genève, 2012. URL: http://doc.rero.ch/record/31218.

[Ecl14] Eclipse. *Atlas Transformation Language*. Acedido em Janeiro de 2014. URL: http://wiki.eclipse.org/ATL/Concepts.

[EBS13] Y. Elkhatib, G. S. Blair e B. Surajbali. "Experiences of Using a Hybrid Cloud to Construct an Environmental Virtual Observatory". Em: *Proceedings of the 3rd International Workshop on Cloud Data and Platforms*. CloudDP '13. Prague, Czech Republic: ACM, 2013, pp. 13–18. ISBN: 978-1-4503-2075-7. DOI: 10.1145/2460756.2460759. URL: http://doi.acm.org/10.1145/2460756.2460759.

[EP] M. Emrich e D. Press. *Behaviour Driven Development with JavaScript: An Introduction to BDD with Jasmine*. Developer Press. ISBN: 9781909264113. URL: http://books.google.pt/books?id=dhtbOtOHM3UC.

[Eng+05] G. Engels, A. F. Örster, R. Heckel e S. T. Öne. "Process Modeling Using UML". Em: *Process-Aware Information Systems*. Wiley, 2005, pp. 85–117.

[EMJ10] H. Erdogmus, G. Melnik e R. Jeffries. "Test-Driven Development". Em: *Encyclopedia of Software Engineering*. 2010, pp. 1211–1229.

[Eva03] Evans. *Domain-Driven Design: Tacking Complexity In the Heart of Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN: 0321125215.

[Fed13] C. Feduke. *Instant RSpec Test - Driven Development How-to*. Packt Publishing, 2013. ISBN: 1782165223, 9781782165224.

[Fou12a] T. E. Foundation. *Example: Generate HTML documentation from an Ecore metamodel with EGL*. Acedido em Janeiro de 2012. URL: http://www.eclipse.org/epsilon/examples/index.php?example=org.eclipse.epsilon.examples.egldoc.

[Fou12b] T. E. Foundation. *Example: Transform a Tree model to a Graph model with ETL - Acedido em 2014*. Acedido em Janeiro de 2012. URL: http://www.eclipse.org/epsilon/examples/index.php?example=org.eclipse.epsilon.examples.egldoc.

[Fow10] M. Fowler. *Domain-Specific Languages*. Addison-Wesley Signature Series (Fowler). Pearson Education, 2010. ISBN: 9780131392809. URL: http://books.google.pt/books?id=ri1muolw%5C_YwC.

[Fra+03] S. Fraser, K. Beck, B. Caputo, T. Mackinnon, J. Newkirk e C. Poole. "Test driven development (TDD)". Em: *Proceedings of the 4th international conference on Extreme programming and agile processes in software engineering*. XP'03. Genova, Italy: Springer-Verlag, 2003, pp. 459–462. ISBN: 3-540-40215-2. URL: http://dl.acm.org/citation.cfm?id=1763875.1763973.

[Fre+05] T. K. Fredericks, S. D. Choi, J. Hart, S. E. Butt e A. Mital. "An investigation of myocardial aerobic capacity as a measure of both physical and cognitive workloads". Em: *International Journal of Industrial Ergonomics* 35.12 (2005), pp. 1097–1107. ISSN: 0169-8141. DOI: http://dx.doi.org/10.1016/j.ergon.2005.06.002. URL: http://www.sciencedirect.com/science/article/pii/S0169814105000971.

[GS09] R. E. Gallardo-Valencia e S. E. Sim. "Continuous and Collaborative Validation: A Field Study of Requirements Knowledge in Agile". Em: *Proceedings of the 2009 Second International Workshop on Managing Requirements Knowledge*. MARK '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 65–74. ISBN: 978-0-7695-4099-3. DOI: 10.1109/MARK.2009.3. URL: http://dx.doi.org/10.1109/MARK.2009.3.

[GK06] S. Garde e P. Knaup. "Requirements Engineering in Health Care: The Example of Chemotherapy Planning in Paediatric Oncology". Em: *Requir. Eng.* 11.4 (ago. de 2006), pp. 265–278. ISSN: 0947-3602. DOI: 10.1007/s00766-006-0029-6. URL: http://dx.doi.org/10.1007/s00766-006-0029-6.

[GZ13] V. Garousi e J. Zhi. "A survey of software testing practices in Canada". Em: *Journal of Systems and Software* 86.5 (2013), pp. 1354–1376. ISSN: 0164-1212. DOI: http://dx.doi.org/10.1016/j.jss.2012.12.051. URL: http://www.sciencedirect.com/science/article/pii/S0164121212003561.

[Gri+08] D. Grimes, D. S. Tan, S. E. Hudson, P. Shenoy e R. P. Rao. "Feasibility and Pragmatics of Classifying Working Memory Load with an Electroencephalograph". Em: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. Florence, Italy: ACM, 2008, pp. 835–844. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357187. URL: http://doi.acm.org/10.1145/1357054.1357187.

[Grö13] M. Gröber. "Investigation of the Usage of Artifacts in Agile Methods". Tese de mestrado. TECHNISCHEN UNIVERSITÄT MÜNCHEN, 2013. URL: http://www4.in.tum.de/~kuhrmann/studworks/mg-thesis.pdf.

[Gro14] T. S. Group. *CHAOS SUMMARY FOR 2010*. Acedido em Janeiro de 2014. URL: http://insyght.com.au/special/2010CHAOSSummary.pdf.

[Haa+10] E. Haapalainen, S. Kim, J. F. Forlizzi e A. K. Dey. "Psycho-physiological Measures for Assessing Cognitive Load". Em: *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*. Ubicomp '10. Copenhagen, Denmark: ACM, 2010, pp. 301–310. ISBN: 978-1-60558-843-8. DOI: 10.1145/1864349.1864395. URL: http://doi.acm.org/10.1145/1864349.1864395.

[Hah13] E. Hahn. *JavaScript Testing with Jasmine: JavaScript Behavior-Driven Development*. O'Reilly Media, Inc., 2013. ISBN: 1449356370, 9781449356378.

[HU12] S. Hammond e D. Umphress. "Test driven development: the state of the practice". Em: *Proceedings of the 50th Annual Southeast Regional Conference*. ACM-SE '12. Tuscaloosa, Alabama: ACM, 2012, pp. 158–163. ISBN: 978-1-4503-1203-5. DOI: 10.1145/2184512.2184550. URL: http://doi.acm.org/10.1145/2184512.2184550.

[HS88] S. G. Hart e L. E. Staveland. "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research". Em: *Human Mental Workload*. Ed. por P. A. Hancock e N. Meshkati. Vol. 52. Advances in Psychology. North-Holland, 1988, pp. 139–183. DOI: http://dx.doi.org/10.1016/S0166-4115(08)62386-9. URL: http://www.sciencedirect.com/science/article/pii/S0166411508623869.

[HW12] A. Hellesoy e M. Wynne. *The Cucumber Book: Behaviour-Driven Development for Testers and Developers*. Pragmatic Programmers. Pragmatic Bookshelf, 2012. ISBN: 9781934356807. URL: http://books.google.pt/books?id=oMswygAACAAJ.

[Hjo+04] N. Hjortskov, D. Rissén, A. Blangsted, N. Fallentin, U. Lundberg e K. Søgaard. "The effect of mental stress on heart rate variability and blood pressure during computer work". English. Em: *European Journal of Applied Physiology* 92.1-2 (2004), pp. 84–89. ISSN: 1439-6319. DOI: 10.1007/s00421-004-1055-z. URL: http://dx.doi.org/10.1007/s00421-004-1055-z.

[Hun06] J. Hunt. *Agile software construction*. Springer, 2006, pp. I–X, 1–254. ISBN: 978-1-85233-944-9.

[Kar12] M. Kart. "Behavior-driven Development: Conference Tutorial". Em: *J. Comput. Sci. Coll.* 27.4 (abr. de 2012), pp. 75–75. ISSN: 1937-4771. URL: http://dl.acm.org/citation.cfm?id=2167431.2167445.

[KC07] B. Kitchenham e S. Charters. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Rel. téc. EBSE 2007-001. Keele University e Durham University Joint Report, 2007.

[Kos07] L. Koskela. *Test driven: practical tdd and acceptance tdd for java developers*. Greenwich, CT, USA: Manning Publications Co., 2007. ISBN: 9781932394856.

[Kot+96] G. Kotonya, G. Kotonya, I. Sommerville e I. Sommerville. "Requirements Engineering With Viewpoints". Em: *Software Engineering Journal* 11 (1996), pp. 5–18.

[KS98] G. Kotonya e I. Sommerville. *Requirements Engineering - Processes and Techniques*. John Wiley & Sons, 1998. URL: http://www.comp.lancs.ac.uk/computing/resources/re/.

[Kou01] N. Koudelia. "Acceptance Test-Driven Development". Tese de mestrado. UNIVERSITY OF JYVÄSKYLÄ DEPARTMENT OF MATHEMATICAL INFORMATION TECHNOLOGY, 201. URL: http://urn.fi/URN:NBN:fi:jyu-201202161200.

[KLK13] M. Kraemer, D. Ludlow e Z. A. Khan. "Domain-Specific Languages For Agile Urban Policy Modelling." Em: *ECMS*. Ed. por W. Rekdalsbakken, R. T. Bye e H. Zhang. European Council for Modeling e Simulation, 2013, pp. 673–680. ISBN: 978-0-9564944-6-7. URL: http://dblp.uni-trier.de/db/conf/ecms/ecms2013.html#KraemerLK13.

[Kra90] A. F. Kramer. *Physiological metrics of mental workload: A review of recent progress*. Rel. téc. ILLINOIS UNIV AT URBANA-CHAMAPAIGN, 1990.

[KW52] W. H. Kruskal e W. A. Wallis. "Use of Ranks in One-Criterion Variance Analysis". Em: *Journal of the American Statistical Association* 47.260 (1952), pp. 583–621. DOI: 10.1080/01621459.1952.10483441. eprint: http://www.tandfonline.com/doi/pdf/10.1080/01621459.1952.10483441. URL: http://www.tandfonline.com/doi/abs/10.1080/01621459.1952.10483441.

[Küh06] T. Kühne. "Matters of (Meta-)Modeling". Em: *Software and System Modeling* 5.4 (2006), pp. 369–385.

[Lap07] P. Laplante. *What Every Engineer Should Know about Software Engineering*. What Every Engineer Should Know. Taylor & Francis, 2007. ISBN: 9781420006742. URL: http://books.google.pt/books?id=pFHYk0KWAEgC.

[Law+01] I. Lawrence J. Prinzel, P. Alan T., F. Frederick G., S. Mark W. e M. Peter J. *Empirical Analysis of EEG and ERPs for Pyschophysiological Adaptive Task Allocation*. Rel. téc. 2001.

[laz+08] H. lazer, J. Dalton, D. Anderson, M. Konrad e S. Shrum. *'CMMI or Agile: Why Not Embrace Both!,' Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Note*. Rel. téc. CMU/SEI-2008-TN-003. East Lansing, Michigan: Software Engineering Institute, fev. de 2008. URL: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8533.

[LMP10] I. Lazr, S. Motogna e B. Pírv. "Behaviour-Driven Development of Foundational UML Components". Em: *Electron. Notes Theor. Comput. Sci.* 264.1 (ago. de 2010), pp. 91–105. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2010.07.007. URL: http://dx.doi.org/10.1016/j.entcs.2010.07.007.

[LT06] J. C. Lee e D. S. Tan. "Using a Low-cost Electroencephalograph for Task Classification in HCI Research". Em: *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*. UIST '06. Montreux, Switzerland: ACM, 2006, pp. 81–90. ISBN: 1-59593-313-1. DOI: 10.1145/1166253.1166268. URL: http://doi.acm.org/10.1145/1166253.1166268.

[Lef11] D. Leffingwell. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. 1st. Addison-Wesley Professional, 2011. ISBN: 0321635841, 9780321635846.

[Lop11] J. Lopes. "Evaluation of Behavior-Driven Development". Tese de mestrado. Delft, Netherlands: TU Delft, 2011. URL: http://repository.tudelft.nl/view/ir/uuid:90323d56-d208-401e-8d3c-44bec4fca4f4/.

[Lóp+13] J. López-Fernández, J. Cuadrado, E. Guerra e J. de Lara. "Example-driven meta-model development". English. Em: *Software Systems Modeling* (2013), pp. 1–25. ISSN: 1619-1366. DOI: 10.1007/s10270-013-0392-y. URL: http://dx.doi.org/10.1007/s10270-013-0392-y.

[LQ10] A. Lucia e A. Qusef. "Requirements Engineering in Agile Software Development". Em: *Journal of Emerging Technologies in Web Intelligence* 2.3 (2010). URL: http://www.ojs.academypublisher.com/index.php/jetwi/article/view/0203212220.

[Mad10] L. Madeyski. *Test-Driven Development: An Empirical Evaluation of Agile Practice*. Foreword by Prof. Claes Wohlin. (Heidelberg, London, New York): Springer, 2010. ISBN: 978-3-642-04287-4. DOI: 10.1007/978-3-642-04288-1. URL: http://www.springer.com/978-3-642-04287-4.

[MV11] I. Mahmud e V. Veneziano. "Mind-mapping: An effective technique to facilitate requirements engineering in agile software development". Em: *Computer and Information Technology (ICCIT), 2011 14th International Conference on*. Dez. de 2011, pp. 157–162. DOI: 10.1109/ICCITechn.2011.6164775.

[MA04] N. Maiden e I. Alexander. *Scenarios, stories, use cases : through the systems development life-cycle*. Chichester: J. Wiley e sons, 2004. ISBN: 0-470-86194-0. URL: http://opac.inria.fr/record=b1105823.

[MJ96] S. Makeig e T.-P. Jung. "Tonic, phasic, and transient {EEG} correlates of auditory awareness in drowsiness". Em: *Cognitive Brain Research* 4.1 (1996), pp. 15–25. ISSN: 0926-6410. DOI: http://dx.doi.org/10.1016/0926-6410(95)00042-9. URL: http://www.sciencedirect.com/science/article/pii/0926641095000429.

[MW47] H. B. Mann e D. R. Whitney. "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other". Em: *The Annals of Mathematical Statistics* 18.1 (mar. de 1947), pp. 50–60. URL: http://dx.doi.org/10.1214/aoms/1177730491.

[Mat+12] M. Mateo, C. Blasco-Lafarga, I. Martínez-Navarro, J. Guzmán e M. Zabala. "Heart rate variability and pre-competitive anxiety in BMX discipline". English. Em: *European Journal of Applied Physiology* 112.1 (2012), pp. 113–123. ISSN: 1439-6319. DOI: 10.1007/s00421-011-1962-8. URL: http://dx.doi.org/10.1007/s00421-011-1962-8.

[MGR07] T. Memmel, F. Gundelsweiler e H. Reiterer. "Agile Human-centered Software Engineering". Em: *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...But Not As We Know It - Volume 1*. BCS-HCI '07. University of Lancaster, United Kingdom: British Computer Society, 2007, pp. 167–175. ISBN: 978-1-902505-94-7. URL: http://dl.acm.org/citation.cfm?id=1531294.1531317.

[Moo09] D. L. Moody. "The x201C;Physicsx201D; of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering". Em: *IEEE Transactions on Software Engineering* 35.6 (2009), pp. 756–779. ISSN: 0098-5589. DOI: http://doi.ieeecomputersociety.org/10.1109/TSE.2009.67.

[Mor+13] P. Morrison, C. Holmgreen, A. Massey e L. Williams. "Proposing Regulatory-Driven Automated Test Suites". Em: *AGILE*. 2013, pp. 11–21.

[Muk+11] S. Mukherjee, R. Yadav, I. Yung, D. P. Zajdel e B. S. Oken. "Sensitivity to mental effort and test–retest reliability of heart rate variability measures in healthy seniors". Em: *Clinical Neurophysiology* 122.10 (2011), pp. 2059–2066. ISSN: 1388-2457. DOI: http://dx.doi.org/10.1016/j.clinph.2011.02.032. URL: http://www.sciencedirect.com/science/article/pii/S1388245711001830.

[Mul92] L. Mulder. "Measurement and analysis methods of heart rate and respiration for use in applied environments". Em: *Biological Psychology* 34.2–3 (1992). Special Issue Cardiorespiratory Measures and thier Role in Studies of Performance, pp. 205–236. ISSN: 0301-0511. DOI: http://dx.doi.org/10.1016/0301-0511(92)90016-N. URL: http://www.sciencedirect.com/science/article/pii/030105119290016N.

[NEČ30] I. NEČAS. "BDD as a Specification and QA Instrument [online]". Diplomová práce. Masarykova univerzita, Fakulta informatiky, 2011 [cit. 2014-01-30]. URL: Dostupn%C3%A9%20z%20WWW%20%3Chttp://is.muni.cz/th/208305/fi_m/%3E.

[Nie12] M. S. Nielsen. "BDD med SpecFlow vs. TDD". Tese de mestrado. Aarhus Universitet, 2012. URL: http://cs.au.dk/fileadmin/site_files/cs/AA_pdf/BDD-MartinSkovNielsen-final.pdf.

[Nor13] D. North. *Introducing BDD*. Acedido em Novembro de 2013. URL: http://dannorth.net/introducing-bdd/.

[Nor14] D. North. *JBehave*. Acedido em Agosto de 2014. URL: http://jbehave.org.

[OMG14] OMG. *MDA Guide Version 1.0*. Acedido em Janeiro de 2014. URL: http://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf.

[Paa+03] F. Paas, J. E. Tuovinen, H. Tabbers e P. W. M. V. Gerven. "Cognitive load measurement as a means to advance cognitive load theory". Em: *Educational Psychologist* 38 (1 2003), pp. 63–71. DOI: 10.1207/S15326985EP3801_8.

[PEM03] F. Paetsch, A. Eberlein e F. Maurer. "Requirements Engineering and Agile Software Development". Em: *Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*. WETICE '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 308–. ISBN: 0-7695-1963-6. URL: http://dl.acm.org/citation.cfm?id=938984.939792.

[Rog11] R. S. M. Rogerio Atem de Carvalho Fernando Luiz de Carvalho e Silva. "Business Language Driven Development: Joining Business Process Models to Automated Tests". Em: *CoRR* (2011). URL: http://www.confenis2011.aau.dk/digitalAssets/31/31415_proceedings---short-papers.pdf.

[RR11] E. Rubin e H. Rubin. "Supporting Agile Software Development Through Active Documentation". Em: *Requir. Eng.* 16.2 (jun. de 2011), pp. 117–132. ISSN: 0947-3602. DOI: 10.1007/s00766-010-0113-9. URL: http://dx.doi.org/10.1007/s00766-010-0113-9.

[RJB05] J. Rumbaugh, I. Jacobson e G. Booch. *The Unified Modeling Language Reference Manual*. 2ª ed. Boston, MA: Addison-Wesley, 2005. ISBN: 978-0-321-24562-5.

[RM05] K. Ryu e R. Myung. "Evaluation of mental workload with a combined measure based on physiological indices during a dual task of tracking and mental arithmetic". Em: *International Journal of Industrial Ergonomics* 35.11 (2005), pp. 991–1009. ISSN: 0169-8141. DOI: http://dx.doi.org/10.1016/j.ergon.2005.04.005. URL: http://www.sciencedirect.com/science/article/pii/S0169814105000806.

[SVV11] M. dos Santos Soares, J. Vrancken e A. Verbraeck. "User requirements modeling and analysis of software-intensive systems". Em: *Journal of Systems and Software* 84.2 (2011), pp. 328–339. ISSN: 0164-1212. DOI: http://dx.doi.org/10.1016/j.jss.2010.10.020. URL: http://www.sciencedirect.com/science/article/pii/S0164121210002876.

[SL13] L. Schoeneman e J. Liu. "Integrating Behavior Driven Development and Programming by Contract". Em: *Computational Science and Its Applications – ICCSA 2013*. Ed. por B. Murgante, S. Misra, M. Carlini, C. Torre, H.-Q. Nguyen, D. Taniar, B. Apduhan e O. Gervasi. Vol. 7975. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 590–606. ISBN: 978-3-642-39639-7. DOI: 10.1007/978-3-642-39640-3_43. URL: http://dx.doi.org/10.1007/978-3-642-39640-3_43.

[Sch04] K. Schwaber. *Agile project management with Scrum*. Redmond, Wash: Microsoft Press, 2004. ISBN: 9780735619937.

[SDM14a] SDMetrics. *eSense(tm) Meters*. Acedido em Agosto de 2014. URL: http://developer.neurosky.com/docs/doku.php?id=esenses_tm.

[SDM14b] SDMetrics. *SDMetrics*. Acedido em Agosto de 2014. URL: http://www.sdmetrics.com.

[SRP09] H. Sharp, H. Robinson e M. Petre. "The role of physical artefacts in agile software development: Two complementary perspectives". Em: *Interact. Comput.* 21.1-2 (jan. de 2009), pp. 108–116. ISSN: 0953-5438. DOI: 10.1016/j.intcom.2008.10.006. URL: http://dx.doi.org/10.1016/j.intcom.2008.10.006.

[Shu+10] F. Shull, G. Melnik, B. Turhan, L. Layman, M. Diep e H. Erdogmus. "What Do We Know about Test-Driven Development?" Em: *IEEE Software* 27.6 (2010), pp. 16–19. ISSN: 0740-7459. DOI: http://doi.ieeecomputersociety.org/10.1109/MS.2010.152.

[Sil+11] T. S. da Silva, A. Martin, F. Maurer e M. S. Silveira. "User-Centered Design and Agile Methods: A Systematic Review." Em: *AGILE*. IEEE Computer Society, 2011, pp. 77–86. ISBN: 978-0-7695-4370-3. URL: http://dblp.uni-trier.de/db/conf/agiledc/agiledc2011.html#SilvaMMS11.

[ŚNS05] J. Śmia Michałand Bojarski, W. Nowakowski e T. Straszak. "Scenario Construction Tool Based on Extended UML Metamodel". Em: *Proceedings of the 8th International Conference on Model Driven Engineering Languages and Systems*. MoDELS'05. Montego Bay, Jamaica: Springer-Verlag, 2005, pp. 414–429. ISBN: 3-540-29010-9, 978-3-540-29010-0. DOI: 10.1007/11557432_31. URL: http://dx.doi.org/10.1007/11557432_31.

[SWD12] M. Soeken, R. Wille e R. Drechsler. "Assisted Behavior Driven Development Using Natural Language Processing". Em: *Proceedings of the 50th International Conference on Objects, Models, Components, Patterns*. TOOLS'12. Prague, Czech Republic: Springer-Verlag, 2012, pp. 269–287. ISBN: 978-3-642-30560-3. DOI: 10.1007/978-3-642-30561-0_19. URL: http://dx.doi.org/10.1007/978-3-642-30561-0_19.

[SW11] C. Solis e X. Wang. "A Study of the Characteristics of Behaviour Driven Development". Em: *Proceedings of the 2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*. SEAA '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 383–387. ISBN: 978-0-7695-4488-5. DOI: 10.1109/SEAA.2011.76. URL: http://dx.doi.org/10.1109/SEAA.2011.76.

[Som10] I. Sommerville. *Software Engineering*. 9ª ed. Harlow, England: Addison-Wesley, 2010. ISBN: 978-0-13-703515-1.

[STJ11] A. G. Sutcliffe, S. Thew e P. Jarvis. "Experience with user-centred requirements engineering." Em: *Requir. Eng.* 16.4 (2011), pp. 267–280. URL: http://dblp.uni-trier.de/db/journals/re/re16.html#SutcliffeTJ11.

[Tae+11] J. Taelman, S. Vandeput, E. Vlemincx, A. Spaepen e S. Van Huffel. "Instantaneous changes in heart rate regulation due to mental load in simulated office work". English. Em: *European Journal of Applied Physiology* 111.7 (2011), pp. 1497–1505. ISSN: 1439-6319. DOI: 10.1007/s00421-010-1776-0. URL: http://dx.doi.org/10.1007/s00421-010-1776-0.

[Tav+10] H. L. Tavares, G. G. Rezende, V. M. dos Santos, R. S. Manhães e R. A. de Carvalho. "A tool stack for implementing Behaviour-Driven Development in Python Language". Em: *CoRR* abs/1007.1722 (2010). URL: http://dblp.uni-trier.de/db/journals/corr/corr1007.html#abs-1007-1722.

[TFS12] L. Teixeira, C. Ferreira e B. S. Santos. "User-centered requirements engineering in health information systems: A study in the hemophilia field". Em: *Computer Methods and Programs in Biomedicine* 106.3 (2012), pp. 160–174. ISSN: 0169-2607. DOI: http://dx.doi.org/10.1016/j.cmpb.2010.10.007. URL: http://www.sciencedirect.com/science/article/pii/S0169260710002695.

[Van01] A. Van Lamsweerde. "Goal-Oriented Requirements Engineering: A Guided Tour". Em: *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*. RE '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 249–. URL: http://dl.acm.org/citation.cfm?id=882477.883624.

[Voe+13] M. Voelter, S. Benz, C. Dietrich, B. Engelmann, M. Helander, L. C. L. Kats, E. Visser e G. Wachsmuth. *DSL Engineering - Designing, Implementing and Using Domain-Specific Languages*. dslbook.org, 2013, pp. 1–558. ISBN: 978-1-4812-1858-0.

[Wan+14] F. Wanderley, A. Silva, J. Araújo e D. S. da Silveira. "SnapMind: A framework to support consistency and validation of model-based requirements in agile development". Em: *IEEE 4th International Model-Driven Requirements Engineering Workshop, MoDRE 2014, 25 August, 2014, Karlskrona, Sweden*. Ed. por A. Moreira, P. Sánchez, G. Mussbacher e J. Araújo. IEEE, 2014, pp. 47–56. ISBN: 978-1-4799-6343-0. DOI: 10.1109/MoDRE.2014.6890825. URL: http://dx.doi.org/10.1109/MoDRE.2014.6890825.

[WFD12] W. M. Watanabe, R. P. M. Fortes e A. L. Dias. "Using Acceptance Tests to Validate Accessibility Requirements in RIA". Em: *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*. W4A '12. Lyon, France: ACM, 2012, 15:1–15:10. ISBN: 978-1-4503-1019-2. DOI: 10.1145/2207016.2207022. URL: http://doi.acm.org/10.1145/2207016.2207022.

[Wes13] D. West. *Water-Scrum-Fall Is The Reality Of Agile For Most Organizations Today - Acedido em 2014*. Acedido em Dezembro de 2013. URL: http://www.cohaa.org/content/sites/default/files/water-scrum-fall_0.pdf.

[Wil02] G. F. Wilson. "An Analysis of Mental Workload in Pilots During Flight Using Multiple Psychophysiological Measures". Em: *The International Journal of Aviation Psychology* 12.1 (2002), pp. 3–18. DOI: 10.1207/S15327108IJAP1201_2. eprint: http://www.tandfonline.com/doi/pdf/10.1207/S15327108IJAP1201_2. URL: http://www.tandfonline.com/doi/abs/10.1207/S15327108IJAP1201_2.

[WS11] M. A. Winget e W. W. Sampson. "Game Development Documentation and Institutional Collection Development Policy". Em: *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries*. JCDL '11. Ottawa, Ontario, Canada: ACM, 2011, pp. 29–38. ISBN: 978-1-4503-0744-4. DOI: 10.1145/1998076.1998083. URL: http://doi.acm.org/10.1145/1998076.1998083.

[WG11] D. Wüest e M. Glinz. "Flexible Sketch-Based Requirements Modeling". Em: *REFSQ*. Ed. por D. M. Berry e X. Franch. Vol. 6606. Lecture Notes in Computer Science. Springer, 2011, pp. 100–105. ISBN: 978-3-642-19857-1.

[ZAS14] H. Zarzour, T. Abid e M. Sellami. "Conflict-Free Collaborative Decision-Making over Mind-Mapping". Em: *Advanced Computing Communication Technologies (ACCT), 2014 Fourth International Conference on*. Fev. de 2014, pp. 509–515. DOI: 10.1109/ACCT.2014.34.

Artigos BDD selecionados

Tabela A.1: Livros, artigos e dissertações utilizados para a questão 2 - *Qual é a adoção do BDD nos processos ágeis?*

#	Nome	Resumo
1	The RSpec Book [Che+10]	Livro da linguagem RSpec
2	The Cucumber Book: Behaviour-Driven Development for Testers and Developers [HW12]	Livro da linguagem Cucumber
3	Behaviour Driven Development with JavaScript - An Introduction to BDD with Jasmine [EP]	Livro do *framework* Jasmine
4	JavaScript Testing with Jasmine: JavaScript Behavior-Driven Development [Hah13]	Livro do *framework* Jasmine
5	Instant RSpec Test-Driven Development How-to [Fed13]	Livro de RSpec
6	BDD as a Specification and QA Instrument [NEČ30]	BDD como técnica de negociação do sistema e controlo de qualidade
7	Behaviour Driven Development and integration platform [Che12]	IDE para gestão de projetos com o BDD
8	Evaluation of Behavior-Driven Development [Lop11]	Avaliação do BDD com caso de estudo
9	Automated testing of non-functional requirements based on behavioural scripts [BE]	BDD para testar Req. NFs
10	BDD med SpecFlow vs. TDD [Nie12]	Desenvolvimento de aplicação para *Windows Phone*
11	Behaviour-Driven Development [Don12]	Análise de *framworks* BDD
12	An agile approach to requirement specification [Ban07]	BDD aplicado para desenvolver uma *intranet*
13	Domain-specific languages for agile urban policy modeling [KLK13]	BDD aplicado num projeto de política urbana
14	Using acceptance tests to validate accessibility requirements in RIA [WFD12]	BDD aplicado em RIA - Rich Client Applications
15	Experiences of using a hybrid cloud to construct an environmental virtual observatory [EBS13]	Desenvolvimento em *Cloud* utilizando BDD
16	Behavior Driven Development for circuit design and verification [Die+12]	BDD aplicado em desenho de circuitos digitais
17	Behaviour Driven Development for Multi-Agent Systems [CIG13]	BDD aplicado em *Multi-Agent Systems* (MAS)
18	Proposing Regulatory-Driven Automated Test Suites [Mor+13]	Casos de testes a sistemas de saúde com BDD
19	Assisted behavior-driven development using natural language processing [SWD12]	Transformar cenários textuais em testes de aceitação
20	Test driven development: the state of the practice [HU12]	Estado da arte do TDD
21	Behaviour Driven Development of Foundational UML Components [LMP10]	Estudo de componentes UML do BDD
22	A study of the characteristics of behaviour driven development [SW11]	Resumo das características do BDD
23	A survey of software testing practices in Canada [GZ13]	*Survey* de práticas de Testes
24	A tool stack for implementing Behaviour-Driven Development in Python Language [Tav+10]	Ferramentas para usar BDD com *Python*
25	Mapping Business Process Modeling constructs to Behavior Driven Development Ubiquitous Language [CCM10]	Ligar Diagramas de Estados com o BDD
26	Towards automatic scenario generation from coverage information [Die+13]	Testes BDD e cobertura de código
27	Integrating Behaviour Driven Development and programing By Contract [SL13]	Combina *Programming By Contract* com BDD
28	The Perfect Couple: Domain Models and Behavior-Driven Development [Dan13]	Melhorar Linguagem do BDD com Modelos de Domínio
29	Documentation by Example [Bro09]	Criar documentação a partir do BDD
30	Implementing Behavior Driven Development in an Open Source ERP [Car+13]	BDD para verificar requisitos num sistema ERP5
31	Envisioning the next-generation of functional testing tools [And07]	Características de ferramentas de testes
32	Behavior-driven development: conference tutorial [Kar12]	Tutorial de BDD
33	Business language driven development: Joining business process models to automated tests [Rog11]	Ligar modelos de processo com os testes BDD

B

Questionários

B.1 Questionário Background Participantes

Background Questionnaire

* Required

1. **Name** *

2. **Gender** *
   Mark only one oval.
   - ◯ Male
   - ◯ Female
   - ◯ Other
   - ◯ Decline to state

3. **Date of birth** *
   Example: December 15, 2012

4. **Years of higher education (university and up)** *

5. **Do you have degrees in (or are taking one of)** *
   Mark only one oval.
   - ◯ Computer Science
   - ◯ Computer Engineering
   - ◯ Software Engineering
   - ◯ Other computing-related field
   - ◯ Other engineering field
   - ◯ Other field

6. **If you selected "Other engineering field" or "Other field" in the previous question, please specify the degree**

7. **Highest degree obtained**

8. **Do you know any Behavioural Models?** *
   Ex. Activity Diagrams, State Diagrams, Sequence Diagrams
   Mark only one oval.
   - ◯ Yes
   - ◯ No

9. **Are you familiar with Behaviour-Driven Development (BDD)?** *
Mark only one oval.

- Yes
- No

B.2 Questionário Feedback Participantes

Final Form

* Required

1. **Name** *

2. **Difficulty: Overall I found the assignment to be…** *
   Mark only one oval.
   - Very difficult
   - Difficult
   - Somewhat difficult
   - Somewhat easy
   - Easy
   - Very Easy

3. **Adequacy: I found the explanations and documentation provided to be…** *
   Mark only one oval.
   - Comprehensive
   - Adequate
   - Inadequate

4. **Enjoyment: I found the assignment to be...** *
   Mark only one oval.
   - Very enjoyable
   - Enjoyable
   - Somewhat enjoyable
   - Somewhat boring
   - Boring
   - Very Boring

5. **Rank each of these problems from easy to difficult by assigning them numbers from 1 to 10. If you think some of the problems are equally difficult, you can assign them the same number.** *
   1 (Easiest), 10 (Hardest)
   Mark only one oval per row.

	1	2	3	4	5	6	7	8	9	10
TP1										
TP2										
C1										
C2										
C3										
C4										
C5										
C6										

6. **Comments**
   About the tool, the practical and comprehension exercises

.....

.....

.....

.....

.....

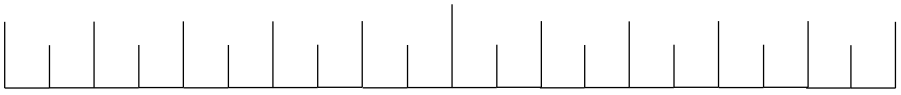
B.3 Questionário NASA-TLX

RATING SHEET

Subject Name: _____ Task ID: _____

Put an "X" on each of the six scales at the point which matches your experience.

PERFORMANCE



Good Poor

MENTAL DEMAND

Low High

PHYSICAL DEMAND

Low High

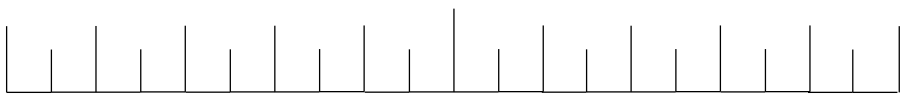
TEMPORAL DEMAND

Low High

EFFORT

Low High

FRUSTRATION



Low High

SOURCES-OF-WORKLOAD COMPARISON BOXES

In order to assess the relative importance of the factors in the rating scale, we ask you to choose the most important factor in a pair of rating scale titles.

In each box circle the scale title that represents the more important contributor to workload for the specific task(s) you performed in this experiment.

Effort or Performance	Temporal Demand or Frustration	Temporal Demand or Effort
Physical Demand or Frustration	Performance or Frustration	Physical Demand or Temporal Demand
Physical Demand or Performance	Temporal Demand or Mental Demand	Frustration or Effort
Performance or Mental Demand	Performance or Temporal Demand	Mental Demand or Effort
Mental Demand or Physical Demand	Effort or Physical Demand	Frustration or Mental Demand

RATING SCALE INSTRUCTIONS

In order to assess the experience you had during the tasks you carried out, we are asking you to fill out a sheet of rating scales. In the most general sense, we are examining the "workload" you experienced. Each line on the rating scale has two end point descriptors that describe the scale. Please put an "X" on each of the six scales at the point which matches your experience. The scales we used are described below. Note that "Performance" scale goes from "good" on the left to "bad" on the right while the other scales go from "low" in the left to "high" on the right. Please consider your responses carefully distinguishing among the different task conditions. Consider each scale individually. Your ratings will play an important role in evaluating our study. Your participation is essential to the success of this experiment and is greatly appreciated by us.

RATING SCALE DEFINITIONS

Title	**Endpoints**	**Descriptions**
PERFORMANCE	*Good/Poor*	How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfies were you with your performance in accomplishing these goals?
MENTAL DEMAND	*Low/High*	How much mental and perceptual activity was required (e.g., thinking, deciding, calculating, remembering, looking, searching, etc.)? Was the task easy or demanding, simple or complex, exacting or forgiving?
PHYSICAL DEMAND	*Low/High*	How much physical activity was required (e.g., pushing, pulling, controlling, activating, holding a position etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?
TEMPORAL DEMAND	*Low/High*	How much time pressure did you feel due to the rate or pace at which the task or tasks elements occurred? Was the pace slow and leisurely or rapid and frantic?
EFFORT	*Low/High*	How hard did you have to work (mentally and physically) to accomplish your level of performance?
FRUSTRATION LEVEL	*Low/High*	How insecure, discouraged, irritated, stressed, and annoyed versus secure, gratified, content, relaxed, and complacent did you feel during the task?

Código das linguagens

C.1 Regras de validação do DomainMap

```
1
// GENERIC MIND MAP RESTRICTIONS
3
context RootNode {
5   constraint OnlyOneRoot {
      check : self.select(f : Node | isCenter(f.type())).size() = 1
7     message: 'There can only be one root type Node in a MindMap Model'
    }
9 }

11 context Node {
    constraint nameNotEmpty {
13    check : self.nodeText.isDefined()

15    message: 'Node "' + self.nodeText + '" must have a name'
    }
17  constraint OnlyOneWord {
      check : self.nodeText.asString().matches("^[a-zA-Z]+$")
19
      message: 'Node "' + self.nodeText + '" must only have one word'
21  }
  }
23
  context MindMapModel {
25  // CHANGED FOR THIS CASE ONLY
    constraint EveryNodeMustHaveAEdge {
```

```
    check {
      var n = Node;
      n = self.nodes.select(n : Node | not n.type().name.equals("Enum") and
          self.edges.select(c: Edge | c.source = n or c.target = n).isEmpty());
      return n.isEmpty();
    }
    message : 'All nodes must have a edge associated: ' +
        transformSetToReadable(n+"")
  }
}

context Edge {
  constraint PointToSelf {
    check {
      var r = self.select(e: Edge | ((e.source <> null  and e.source.nodeText
          <> null) and e.source.nodeText.equals(e.target.nodeText) and e.source
          .type().name.equals(e.target.type().name)));
      return r.size() == 0;
    }
    message : 'Nodes cannot connect to themselves: ' + transformSetToReadable(r
        .first.target+"")
  }
}

// DOMAIN MODEL MIND MAP RESTRICTIONS

context MindMapModel {
  constraint AttributesCanOnlyPointToEntity {
    check {
      var n = self.nodes.select(n : Node | isAttribute(n.type()) and
        self.edges.select(c: Edge | c.target = n and (isEntity(c.source.type())
            or isAssociativeEntity(c.source.type()))).isEmpty());
      return n.isEmpty();
    }
    message : 'Atributes can only link to Entities: ' + transformSetToReadable(
        n+"")
  }

  constraint EntityMustPointToRootNodeOrEntities {
    check {
      var n = Node;
      n = self.nodes.select(n : Node | isEntity(n.type()) and self.edges.select
          (c: Edge | (isCenter(c.source.type()) or isEntity(c.source.type()) or
           isAssociativeEntity(c.source.type())) and c.target = n).isEmpty());
      return n.isEmpty();
    }
    message : 'Entities must point to Domain or Entities Node"s: ' +
        transformSetToReadable(n+"")
  }
  constraint AnAssociatedAttributeMustAssociations {
```

```
    check {
      var firstAssociation = self.nodes.select(n : AssociationEntity |
          isAssociativeEntity(n.type()) and
        not self.nodes.select(e : Entity | isEntity(e.type()) and n.
            associations.first.nodeText.equals(e.nodeText)).isEmpty()).size() >
             0;

      var secondAssociation = self.nodes.select(n : AssociationEntity |
          isAssociativeEntity(n.type()) and
        not self.nodes.select(e : Entity | isEntity(e.type()) and n.
            associations.last.nodeText.equals(e.nodeText)).isEmpty()).size() >
            0;

      return firstAssociation and secondAssociation;
    }
    message : 'The associations of an Association Entity must be defined '
  }
  constraint ThereCanBeOnlyOneDefinitionOfEntityForEachParentNode {
    check {
      var root_sons = self.edges.select(e: Edge | isCenter(e.source.type()));
      var f = true;
      var badNode = "";
      var entitiesToParse = new Sequence;
      badNode = checkIfOnlyOneDefinition(root_sons);
      if (badNode <> null) {
        return false;
      } else {
        var entities = self.nodes.select(n : Node | isEntity(n.type()));
        for (e in entities) {
          var sons = self.edges.select(ee: Edge | ee.source = e);
          badNode = checkIfOnlyOneDefinition(sons);
          if (badNode <> null) { return false; }
        }
      }
      return f == true;
    }
    message : 'Only one Entity definition per node is allowed: ' +
        transformSetToReadable(badNode+"")
  }
}
//No Enumerate attribute can be associated
context Edge {
  constraint NoNodePointToEnumerate {
    check {
      var r = self.select(e: Edge | ( isEnum(e.target.type())));
      return r.size() == 0;
    }
    message : 'Nodes cannot point to Enumeration nodes: ' +
        transformSetToReadable(r.first.source+"")
  }
```

```
109   constraint EnumerateNodesCanOnlyPointToLeafNodes {
        check {
111       var r = self.select(e: Edge | (isEnum(e.source.type()) and not isConstant
              (e.target.type())));
          return r.size() == 0;
113     }
        message : 'Enumeration nodes can only connect to Constants nodes: ' +
            transformSetToReadable(r.first.target+"")
115   }
    }
117
    // OPERATIONS
119
    operation Any isCenter(c : EClass) : Boolean {
121   if (c.isUndefined()) return false;
      if (c.name.equals("RootNode")) {
123     return true;
      } else {
125     return false;
      }
127 }
    operation Any isEntity(c : EClass) : Boolean {
129   if (c.isUndefined()) return false;
      if (c.name.equals("Entity")) {
131     return true;
      } else {
133     return false;
      }
135 }
    operation Any isAssociativeEntity(c : EClass) : Boolean {
137   if (c.isUndefined()) return false;
      if (c.name.equals("AssociationEntity")) {
139     return true;
      } else {
141     return false;
      }
143 }
    operation Any isAttribute(c : EClass) : Boolean {
145   if (c.isUndefined()) return false;
      if (c.name.equals("Attribute")) {
147     return true;
      } else {
149     return false;
      }
151 }
    operation Any isConstant(c : EClass) : Boolean {
153   if (c.isUndefined()) return false;
      if (c.name.equals("Constant")) {
155     return true;
      } else {
```

```
157       return false;
    }
159 }
operation Any isEnum(c : EClass) : Boolean {
161   if (c.isUndefined()) return false;
    if (c.name.equals("Enum")) {
163     return true;
    } else {
165     return false;
    }
167 }
operation Any transformSetToReadable(set : String ) : String {
169   var r = set.split("nodeText=");
    var result : String;
171   var i = 0;
    for (s in r) {
173     if (i > 0) {
        if (result.isEmpty()) {
175         result = s.split(",")[0];
        } else {
177         result = result + ", " + s.split(",")[0];
        }
179     }
      i = i + 1;
181   }
    return result.trim();
183 }
operation Any checkIfOnlyOneDefinition(nodes : Collection) : Entity {
185   var parsedEntities = new Map;
    var badNode = null;
187   for (e in nodes) {
      if (parsedEntities.containsKey(e.target.nodeText)) {
189       badNode = e.target;
        break;
191     }
      parsedEntities.put(e.target.nodeText,"");
193   }
    return badNode;
195 }
```

C.2 Regras de validação do BehaviorMap

```
1 // ----------------------------------
  //
3 // GENERIC MIND MAP RESTRICTIONS
  //
5 // ----------------------------------
  context RootNode {
7   constraint OnlyOneRoot {
```

```
    check : self.select(f : Node | isCenter(f.type())).size() = 1
    message: 'There can only be one root type Node in a MindMap Model'
  }
}

context Node {
  constraint nameNotEmpty {
    check : self.nodeText.isDefined()

    message: 'Node "' + self.nodeText + '" must have a name'
  }
  constraint OnlyOneWord {
    check : self.nodeText.asString().split("\\ ").size() > 0

    message: 'Node "' + self.nodeText + '" must only have one word'
  }
}

context MindMapModel {
  // CHANGED FOR THIS CASE ONLY
  constraint EveryNodeMustHaveAEdge {
    check {
      var n = Node;
      n = self.nodes.select(n : Node | not n.type().name.equals("Enum") and
          self.edges.select(c: Edge | c.source = n or c.target = n).isEmpty());
      return n.isEmpty();
    }
    message : 'All nodes must have a edge associated: ' +
        transformSetToReadable(n+"")
  }
}

context Edge {
  constraint PointToSelf {
    check {
      var r = self.select(e: Edge | ((e.source <> null  and e.source.nodeText
          <> null) and e.source.nodeText.equals(e.target.nodeText) and e.source
          .type().name.equals(e.target.type().name)));
      return r.size() == 0;
    }
    message : 'Nodes cannot connect to themselves: ' + transformSetToReadable(r
        .first.target+"")
  }
}
// ---------------------------------
//
// BEHAVIORMAP RESTRICTIONS
//
// ---------------------------------
context MindMapModel {
```

```
53  constraint GivenHasToHaveAtLeastOneEntity {
      check {
55      var n = self.nodes.select(n : Node | isGiven(n.type()) and
          self.edges.select(c: Edge | c.source = n and (isEntity(c.target.type())
              )).isEmpty());
57      return n.isEmpty();
      }
59    message : 'Step Given must have at least one Entity'
    }
61
    constraint AgentMustPointToActions {
63    check {
        var n = self.nodes.select(n : Node | isAgent(n.type()) and
65        self.edges.select(c: Edge | c.source = n and not (isAction(c.target.
              type()))).size() > 0);
        return n.isEmpty();
67    }
      message : 'Agent can only link to Actions: ' + transformSetToReadable(n+"")
69  }

71  constraint GivenMustPointToEntities {
      check {
73      var n = self.nodes.select(n : Node | isGiven(n.type()) and
          self.edges.select(c: Edge | c.source = n and not (isEntity(c.target.
              type()))).size() > 0);
75      return n.isEmpty();
      }
77    message : 'Step Given can only link to Entities: ' + transformSetToReadable
          (n+"")
    }
79
    constraint CheckIfGivenAsAnyActions {
81    check {
        var given = self.nodes.select(n : Node | isGiven(n.type())).first;
83      return not checkIfHasActions(given);
      }
85    message : 'Step Given cannot have any actions'
    }
87
    constraint CheckIfWhenAsAnyActions {
89    check {
        var when = self.nodes.select(n : Node | isWhen(n.type())).first;
91      return checkIfHasActions(when) or checkIfHasValues(when);
      }
93    message : 'Step when must have at least one action or an attribute
          definition'
    }
95
    constraint CheckIfThenAsAnyActionsOrEntity {
97    check {
```

```
      var then = self.nodes.select(n : Node | isThen(n.type())).first;
      var hasAgent = false;
      for (e in then.entities) {
        if (isAgent(e.type())) {
          hasAgent = true;
          break;
        }
      }
      if (hasAgent) {
        return checkIfHasActions(then);
      } else {
        return checkIfHasActions(then) or then.entities.size() > 0;
      }
    }
    message : 'Step then must have at least one action or an entity defined'
  }

  constraint CheckIfThereIsEntitiesWithTheSameNameInSameStep {
    check {
      var given = self.nodes.select(n : Node | isGiven(n.type())).first;
      var when = self.nodes.select(n : Node | isWhen(n.type())).first;
      var then = self.nodes.select(n : Node | isThen(n.type())).first;
      return not searchForEqualNames(given.entities) and
        not searchForEqualNames(when.entities) and
        not searchForEqualNames(then.entities);
    }
    message : 'Every entity"s name must be unique within the same step'
  }

  constraint CheckIfWhenActions {
    check {
      var when = self.nodes.select(n : Node | isWhen(n.type())).first;
      var ent = getActions(when.entities);
      if (ent == null) {
        return true;
      }
      return false;
    }
    message : 'There is an entity in Step When that is malformed: ' + ent.
        nodeText
  }

  critique checkIfEntitysThenAsValues {
    check {
      var then = self.nodes.select(n : Node | isThen(n.type())).first;
      return not checkIfHasValues(then);
    }
    message : 'Attribute must be defined in the step Then in order to test
        anything'
  }
```

```
147   critique checkIfThenActionAsReturn {
        check {
149       var then = self.nodes.select(n : Node | isThen(n.type())).first;
          return searchForActionsReturn(then.entities);
151     }
        message : 'Returns must be defined in the step Then in order to test
            anything'
153   }
}
155
context Node {
157   critique EntityMustHaveType {
        check {
159       if (isEntity(self.type())) {
            return not (self.correspond == null);
161       } else { return true; }
        }
163     message : 'Entities must have a type: ' + self.nodeText
    }
165 }

167 // CUSTOM FUNCTIONS

169 operation Any getActions(nodes : Collection) : Any {
    var toParse = nodes;
171 for (node in toParse) {
      if (node.actions.size() > 0) {
173     var node_actions = node.actions;
        for (node_action in node_actions) {
175       for(ent in node_action.entityPar) {
            if (ent.attributes.size() > 0 and ent.represents <> null) {
177           return ent;
            } else if (ent.attributes.size() == 0 and ent.represents == null) {
179           return ent;
            }
181       }
        }
183   }
      toParse.addAll(node.entities);
185 }
    return null;
187 }

189 operation Any searchForActionsReturn(nodes : Collection) : Boolean {
    var toParse = nodes;
191 for (node in toParse) {
      if (node.actions.size() > 0) {
193     var node_actions = node.actions;
        for (node_action in node_actions) {
```

```
195         if ((node_action.entityPar.size() = 0) or (node_action.valuePar.size()
                = 0)) {
              return true;
197         }
          }
199     }
      toParse.addAll(node.entities);
201   }
    return false;
203 }

205 operation Any checkIfHasActions(step : Node) : Boolean {
    var steps_sons = step.entities;
207   return searchForActions(steps_sons);
  }
209
  operation Any checkIfHasValues(step : Node) : Boolean {
211   var steps_sons = step.entities;
    return searchForValue(steps_sons);
213 }

215 operation Any searchForValue(nodes : Collection) : Boolean {
    var toParse = nodes;
217   for (node in toParse) {
      var entity_attribute = node.attributes;
219     for (att in entity_attribute) {
        if (att.value.isDefined()) {
221       return true;
        }
223     }
      toParse.addAll(node.entities);
225   }
    return false;
227 }

229 operation Any searchForActions(nodes : Collection) : Boolean {
    var toParse = nodes;
231   for (node in toParse) {
      if (node.actions.size() > 0) {
233     return true;
      }
235   toParse.addAll(node.entities);
    }
237   return false;
  }
239
  operation Any searchForEqualNames(nodes : Collection) : Boolean {
241   var toParse = nodes;
    var names = new Map;
243   for (node in toParse) {
```

```
    if (names.containsKey(node.nodeText)) {
      return true;
    } else {
      names.put(node.nodeText,node.nodeText);
    }
    toParse.addAll(node.entities);
  }
  return false;
}


operation Any transformSetToReadable(set : String ) : String {
  var r = set.split("nodeText=");
  var result : String;
  var i = 0;
  for (s in r) {
    if (i > 0) {
      if (result.isEmpty()) {
        result = s.split(",")[0];
      } else {
        result = result + ", " + s.split(",")[0];
      }
    }
    i = i + 1;
  }
  return result.trim();
}

operation Any isCenter(c : EClass) : Boolean {
  if (c.isUndefined()) return false;
  if (c.name.equals("RootNode")) {
    return true;
  } else {
    return false;
  }
}
operation Any isEntity(c : EClass) : Boolean {
  if (c.isUndefined()) return false;
  if (c.name.equals("Entity")) {
    return true;
  } else {
    return false;
  }
}
operation Any isAssociativeEntity(c : EClass) : Boolean {
  if (c.isUndefined()) return false;
  if (c.name.equals("AssociationEntity")) {
    return true;
  } else {
    return false;
```

```
  }
}
operation Any isAttribute(c : EClass) : Boolean {
  if (c.isUndefined()) return false;
  if (c.name.equals("Attribute")) {
    return true;
  } else {
    return false;
  }
}

operation Any isAction(c : EClass) : Boolean {
  if (c.isUndefined()) return false;
  if (c.name.equals("Action")) {
    return true;
  } else {
    return false;
  }
}

operation Any isAgent(c : EClass) : Boolean {
  if (c.isUndefined()) return false;
  if (c.name.equals("Agent")) {
    return true;
  } else {
    return false;
  }
}

operation Any isGiven(c : EClass) : Boolean {
  if (c.isUndefined()) return false;
  if (c.name.equals("Given")) {
    return true;
  } else {
    return false;
  }
}

operation Any isWhen(c : EClass) : Boolean {
  if (c.isUndefined()) return false;
  if (c.name.equals("When")) {
    return true;
  } else {
    return false;
  }
}

operation Any isThen(c : EClass) : Boolean {
  if (c.isUndefined()) return false;
  if (c.name.equals("Then")) {
```

```
    return true;
  } else {
    return false;
  }
}
```

C.3 Código de Transformação de BehaviorMap para JUnit

```
[%
import "../Convections/JavaConvections.egl";
import "../Convections/BaseConvections.egl";

var initialEntities : Map;

var createdVars : Map;
var treeVars : Map;

out.println(printTestClass(createdVars, treeVars));

operation printTestClass(createdVars : Map, treeVars : Map) {
  var instanciatedEntities : Sequence;

  var worldDefEntities : Sequence;
  var worldEntities : Sequence;
  var worldAsserts : Sequence;

  var thenDefEntities : Sequence;
  var thenEntities : Sequence;
  var thenAsserts : Sequence;

  var imports : Sequence;

  addImports(imports);
  var result = "package test;" + "\n\n";
  result = result + printImports(imports) + "\n";
  result = result + printJavaClass(0, "public", MMBDD!RootNode.allInstances().
      first.nodeText+"Test", null) + " {\n\n";

  var when = printWhen(MMBDD!When.allInstances().first, worldDefEntities,
      worldEntities, worldAsserts, null);
  var given = printGivenWorld(MMBDD!Given.allInstances().first,worldDefEntities
      , worldEntities, worldAsserts);
  var then = printThen(MMBDD!Then.allInstances().first, worldDefEntities,
      thenDefEntities, thenEntities, thenAsserts);

  //worldDefEntities.addAll(worldDefEntities);

  result = result + printWorldEntities(worldDefEntities);
```

```
38    when = printWhen(MMBDD!When.allInstances().first, worldDefEntities,
          worldEntities, worldAsserts, thenDefEntities); //to get the right
          entities there

40    result = result + given;
      result = result + when;
42    result = result + then;

44    return  result + "}";
  }
46  operation addImports(imports : Sequence) {
      imports.add("import org.hamcrest.CoreMatchers;");
48    imports.add("import org.junit.Before;");
      imports.add("import org.junit.Rule;");
50    imports.add("import org.junit.Test;");
      imports.add("import org.junit.rules.ErrorCollector;");
52  }
  operation printImports(imports : Sequence) {
54    var result;
      for(i in imports) {
56      result = result + i;
        result = result + "\n";
58    }
      return result;
60  }
  operation printWorldEntities(entities : Sequence) {
62    var result = "\t";
      var specified : Sequence;
64    result = result + printErrorCollector() + "\n\n\t";
      result = result + printAgent() + "\n\n\t";
66    for (e in entities) {
        if (specified.contains(e.nodeText) or e.isTypeOf(Agent)) { continue; }
68      result = result + printEntityVar(e);
        specified.add(e.nodeText);
70      if (hasMore) { result = result + "\n\t";}
      }
72    return result+"\n\n";
  }
74  operation printErrorCollector() {
      var result = "@Rule\n";
76    result = result + "\tpublic ErrorCollector collector = new ErrorCollector()
          ;";
      return result;
78  }
  operation printAgent() {
80    var result = "private";
      var agent = MMBDD!Agent.allInstances().first.nodeText;
82    return result + " " + agent.firstToUpperCase() + " " + printVariableName(
          agent) + " = new " + agent.firstToUpperCase()+"();";
  }
```

```
84  operation printEntityVar(e : Entity) {
      if (e.correspond == null) {
86      return printJavaVariableDefinition(0, "private","\\\\TODO",
            printVariableName(e.nodeText), null);
      }
88    return printJavaVariableDefinition(0, "private", e.correspond.nodeText,
        printVariableName(e.nodeText), null);
    }
90  operation printEntityInst(e : Entity) {
      if (e.correspond == null) {
92      return printVariableName(e.nodeText) + " = new \\\\TODO";
      }
94    return printVariableName(e.nodeText) + " = new " + e.correspond.nodeText.
        firstToUpperCase() + "();";
    }
96  operation printGivenWorld(step : MMBDD!Given,givenDefEntities : Sequence,
        givenEntities : Sequence, givenAsserts : Sequence) {
      var result;
98    var content : Sequence;
      var exclude : Sequence;
100   var entitiesToInstanciate : Sequence;
      for (e in step.entities) {
102     createInit(e, givenDefEntities, givenEntities, givenAsserts, exclude, false
            );
      }
104
      for(e in givenDefEntities) {
106     if (e.represents == null) { //para so instanciar entidades vinda de
            a c   e s
        content.add(printEntityInst(e));
108     }
        if (not hasMore) {content.add("");}
110   }
      content.addAll(givenEntities);
112   content.add("/*ASSERTIONS*/");
      content.addAll(givenAsserts);
114   content.add("/*--*/");
      var result = printJavaMethodDefinition(1, "public","void", "given", null,
        null, null, content);
116   return "\t@Before" + "\n" + result;
    }
118
    operation printWhen(step : MMBDD!When, givenDefEntities : Sequence,
        givenEntities : Sequence, givenAsserts : Sequence, thenEntities : Sequence)
         {
120   var result;

122   var content : Sequence;
      var whenEntities : Sequence;
124   var whenActions : Sequence;
```

```
  whenActions.add("/*The nexts functions need to be implemented in order to
      have the expected outcome*/");
  var entitiesToParce : Sequence;
  entitiesToParce.addAll(step.entities);
  while (true) {
    if (entitiesToParce.isEmpty()) { break; }
    var e = entitiesToParce.removeAt(0);
    var whenAction = "when" + e.nodeText.firstToUpperCase() + createWhenAction(
        e, whenEntities, givenDefEntities, givenEntities, givenAsserts);
    content.addAll(whenEntities);
    result = result + printJavaMethodDefinition(1, "public","void", whenAction,
         null, null, null, content);
    whenActions.add(printJavaMethodCalling(0,whenAction, null));
    content.clear();
    whenEntities.clear();
    entitiesToParce.addAll(e.entities);
  }
  whenActions.add("/* -- */");
  whenActions.add("/* The nexts variables were created here because they are
      probably related to the previous actions*/");
  for (e in thenEntities) {
    if (e <> null and not e.isTypeOf(Agent) and e.represents == null) {
        whenActions.add(printEntityInst(e)); }
  }
  whenActions.add("/* -- */");
  whenActions.add(printJavaMethodCalling(0,"then", null));
  var mainWhenFunction = printJavaMethodDefinition(1,"public", "void", "when",
      null,null, null, whenActions);
  return "\t@Test" + "\n" + mainWhenFunction + result;
}
operation createWhenAction(e : Entity, whenEntities : Sequence,
    givenDefEntities : Sequence, givenEntities : Sequence, givenAsserts :
    Sequence) {
  var result = "";
  var exclude : Sequence;
  var whenAsserts : Sequence;
  var whenNewEntities : Sequence;

  for (action in e.actions) {
    var pars : Sequence;
    for (p in action.entityPar) { //add EntityPars
      pars.add(printVariableName(p.nodeText));
      createInit(p,givenDefEntities, givenEntities, givenAsserts, exclude,
          false);
    }
    for (p in action.valuePar) { //add ValuePars
      pars.add(printVariableName(p.nodeText));
    }
    var ac = printJavaMethodCalling(0, printVariableName(e.nodeText) + "." +
        printFunctionName(action.nodeText), pars);
```

```
    whenEntities.add(ac);
    result = result + action.nodeText.firstToUpperCase();
    if(hasMore) { result = result + "_And_"; }
  }
  if (e.attributes.size > 0) {
    whenAsserts.clear();
    whenNewEntities.clear();
    createInit(e,givenDefEntities, whenNewEntities, whenAsserts, exclude, false
        );
    whenEntities.addAll(whenNewEntities);
  }
  return result;
}
operation printThen(step : MMBDD!Then, worldDefEntities : Sequence,
    thenDefEntities : Sequence, thenEntities : Sequence, thenAsserts : Sequence
    ) {

  var result;

  var content : Sequence;
  var exclude : Sequence;

  var thenActions : Sequence;

  var functionName : String;

  var toParse = step.entities;
  while(true) {
    if (toParse.isEmpty()) { break; }
    var e = toParse.removeAt(0);
    if (e.entities <> null) {
      if (e.entities.size > 0) { toParse.addAll(e.entities); }
    }
    thenAsserts.clear();
    result = result + createThenAssert(e, thenDefEntities, thenEntities,
        thenAsserts,content);
    thenAsserts.clear();
    createInit(e, thenDefEntities, thenEntities, thenAsserts, exclude, true);
    content.addAll(thenAsserts);
    thenAsserts.clear();
    if (e.represents <> null) {
      content.add("/*Check if all remains the same*/");
      createInit(e.represents, thenDefEntities, thenEntities, thenAsserts,
          exclude, false);
      content.addAll(thenAsserts);
      content.add("/* -- */");
    }
    content.add("");
  }
  //only left new entities mentioned in the then step
```

```
  var toRemove : Sequence;
  for (e in thenDefEntities) {
    for (e2 in worldDefEntities) {
      if (e.nodeText.equals(e2.nodeText)) {
        toRemove.add(e);
        break;
      }
    }
  }
  for (trem in toRemove) {
    thenDefEntities.remove(trem);
  }
  worldDefEntities.addAll(thenDefEntities); //the new entities; the ones that
      are created
  // ------

  var mainMethod = printJavaMethodDefinition(1, "public","void", "then", null,
      null, null, content);
  return mainMethod + result;
}

operation createThenAssert(e : Entity, thenDefEntities : Sequence, thenEntities
    : Sequence, thenAsserts : Sequence, thenContent : Sequence) {
  var result = "";
  var entityName = printVariableName(e.nodeText);
  var functionExclude : Sequence;
  var functionContent : Sequence;
  var functionAsserts : Sequence;
  var functionName = "";

  for (action in e.actions) {
    functionContent.clear();
    functionName = entityName + "Must" + action.nodeText;
    thenContent.add(printJavaMethodCalling(0, functionName, null));
    if (action.entityPar.size() > 1) {
      functionContent.add("/*REFINEMENT IS NEEDED BECAUSE ACTION WITH MULTIPLE
          RETURNS*/");
    }
    for (p in action.entityPar) {
      functionAsserts.clear();
      var entityType = "";
      if (p.correspond <> null) {
        entityType = p.correspond.nodeText;
      }

      functionContent.add(printJavaVariableDefinitionAndInit(0, "", entityType,
           p.nodeText, null, entityName + "." + printFunctionName(action.
          nodeText)+"()"));
      createInit(p,thenDefEntities, thenEntities, functionAsserts,
          functionExclude, true);
```

```
      functionContent.addAll(functionAsserts);
254
      if (p.represents <> null) {
256     functionContent.add("/*Check if all remains the same*/");
        createInit(p.represents, thenDefEntities, thenEntities, thenAsserts,
            functionExclude, false);
258     functionContent.addAll(thenAsserts);
        functionContent.add("/* -- */");
260   }
      thenAsserts.clear();
262 }
    for (p in action.valuePar) {
264   var type = "String";
      var pName = p.nodeText.toLowerCase().replace("\"",'');
266   pName = pName.replace(" ",'_');
      pName = pName.replace("[^a-zA-Z0-9_]",'');
268   pName = "_" + pName;
      if (p.nodeText.matches("[0-9]* *(\\))")) {
270     type = "int";
      } else if (p.nodeText.matches("[0-9]*\\.[0-9]* *(\\))")) {
272     type = "double";
      }
274   functionContent.add(printJavaVariableDefinitionAndInit(0, "private", type
          , pName, null, entityName + "." + printFunctionName(action.nodeText)
          +"()"));
      functionContent.add(createAssertion(p.nodeText, printVariableName(pName))
          );
276 }
    result = result + printJavaMethodDefinition(1, "public","void",
        functionName, null, null, null, functionContent);
278 }
  return result;
280 }

282 operation createInit(e : Entity, stepDefEntities : Sequence, stepEntitiesAction
      : Sequence, stepAsserts : Sequence, exclude : Sequence, toExclude :
    Boolean) { // init a given entity
  var entityName = printVariableName(e.nodeText);
284 var entityType = "entity"; //e.correspond.nodeText;
  var entityHeader = e;
286 stepDefEntities.add(entityHeader);
  for (ent in e.entities) {
288   if (exclude.contains(ent.nodeText)) { continue; }
    var pars : Sequence;
290 var entName = printVariableName(ent.nodeText);
    pars.add(entName);
292 var set = printJavaMethodCalling(0, entityName + "." + printFunctionName("
        set" + entName.firstToUpperCase()), pars);
    var assertSet = createAssertion(entName, entityName + "." +
        printFunctionName("get" + entName.firstToUpperCase())+"()");
```

```
294     createInit(ent,stepDefEntities, stepEntitiesAction, stepAsserts, exclude,
            toExclude);
        stepEntitiesAction.add(set);
296     stepAsserts.add(assertSet);
        if (toExclude) { exclude.add(ent.nodeText); }
298   }

300   for(att in e.attributes) {
        if (exclude.contains(e.nodeText+att.nodeText)) { continue; }
302     var pars : Sequence;
        pars.add(att.value.nodeText);
304     var set = printJavaMethodCalling(0, entityName + "." + printFunctionName("
            set" + att.nodeText.firstToUpperCase()), pars);
        var assertSet = createAssertion(att.value.nodeText, entityName + "." +
            printFunctionName("get" + att.nodeText.firstToUpperCase())+"()");
306     stepEntitiesAction.add(set);
        stepAsserts.add(assertSet);
308     if (toExclude) { exclude.add(e.nodeText+att.nodeText); }
    }
310   stepEntitiesAction.add("");
    stepAsserts.add("");
312 }

314 operation createAssertion(expectation : String, reality : String) {
    var result = "";
316   result = "collector.checkThat(" + expectation + "," + "CoreMatchers.equalTo("
          + reality + "));";
    return result;
318 }
    %]
```

C.4 Código de Transformação de DomainMap para Java

```
1   [%
    import "Convections/JavaConvections.egl";
3   import "MMTypeInference.egl";

5   out.print(printNode(Entity, " ", mappedEnums));
    %]
7
    [%
9   operation Any printFunctions(node : MMClass!Node, mappedEnums : Map,
        VAR_TYPE_BY_INFERECE : Map) : String {
    var functions = "";
11  var parameters : Sequence;
    var parametersType : Sequence;
13  var content : Sequence;
    var varType = "String";
15  for (e in node.attributes) {
```

```
    if (mappedEnums.containsKey(e.nodeText)) {
17    varType = printClassName(e.nodeText);
    } else if (VAR_TYPE_BY_INFERECE.containsKey(e.nodeText.toLowerCase())) {
19    varType = VAR_TYPE_BY_INFERECE.get(e.nodeText.toLowerCase());
    }
21  parameters.add("value"); parametersType.add(varType); content.add(
        printVariableName(e.nodeText) + " = " + "value" + ";");
    functions = functions + printJavaMethodDefinition(1, "public", "void", "set
        "+e.nodeText.firstToUpperCase(), null, parameters, parametersType,
        content);
23
    parameters.clear(); parametersType.clear(); content.clear();
25
    content.add("return " + printVariableName(e.nodeText) + ";");
27  functions = functions + printJavaMethodDefinition(1, "public", varType, "
        get"+e.nodeText.firstToUpperCase(), null, parameters, parametersType,
        content);

29  parameters.clear(); parametersType.clear(); content.clear();
  }
31 for (e in node.entities) {
    parameters.clear(); parametersType.clear(); content.clear();
33  parameters.add("value"); parametersType.add(e.nodeText); content.add(
        printVariableName(e.nodeText) + " = " + "value" + ";");

35  functions = functions + printJavaMethodDefinition(1, "public", "void", "set
        "+e.nodeText.firstToUpperCase(), null, parameters, parametersType,
        content);

37  parameters.clear(); parametersType.clear(); content.clear();

39  content.add("return "+ printVariableName(e.nodeText) + ";");
    functions = functions + printJavaMethodDefinition(1, "public", e.nodeText,
        "get"+e.nodeText.firstToUpperCase(), null, parameters, parametersType,
        content);
41 }
  if (node.type().name.equals("AssociationEntity")) {
43  for (e in node.associations) {
      parameters.clear(); parametersType.clear(); content.clear();
45    parameters.add("value"); parametersType.add(e.nodeText); content.add(
          printVariableName(e.nodeText) + " = " + "value" + ";");

47    functions = functions + printJavaMethodDefinition(1, "public", e.nodeText
          , "set"+e.nodeText.firstToUpperCase(), null, parameters,
          parametersType, content);

49    parameters.clear(); parametersType.clear(); content.clear();

51    content.add("return "+ printVariableName(e.nodeText) + ";");
```

```
      functions = functions + printJavaMethodDefinition(1, "public", e.nodeText
          , "get"+e.nodeText.firstToUpperCase(), null, parameters,
          parametersType, content);
    }
  }
  return functions;
}
operation Any printAttributes(node : MMClass!Node, connector : String,
    mappedEnums : Map, VAR_TYPE_BY_INFERECE : Map) : String {
  var attributes = "";
  var varType = "";
  for (e in node.attributes) {
    varType = "String";
    if (mappedEnums.containsKey(e.nodeText)) {
      varType = printClassName(e.nodeText);
    } else if (VAR_TYPE_BY_INFERECE.containsKey(e.nodeText.toLowerCase())) {
      varType = VAR_TYPE_BY_INFERECE.get(e.nodeText.toLowerCase());
    }
    var comments = e.description;
    attributes = attributes + printJavaVariableDefinition(1, "private", varType
        , e.nodeText, comments) + "\n";
  }
  for (e in node.entities) {
    var comments = e.description;
    attributes = attributes + printJavaVariableDefinition(1, "private", e.
        nodeText, e.nodeText, comments) + "\n";
  }
  if (node.type().name.equals("AssociationEntity")) {
    for (e in node.associations) {
      var comments = e.description;
      attributes = attributes + printJavaVariableDefinition(1, "private", e.
          nodeText, e.nodeText, comments) + "\n";
    }
  }
  return attributes;
}
operation Any printNode(node : MMClass!Node, connector : String, dic : Map) :
    String {
  var result = "";
  --var description = printDescription(node.description);

  var VAR_TYPE_BY_INFERECE : Map;
  if (node.oclRules <> null) {
    for (e in node.oclRules.split("\\n")) {
      parseOCL(e,VAR_TYPE_BY_INFERECE);
    }
  }

  var headerComments = createFileComment(node.nodeText, "1.0", "MMClass", "
      Property of FCT-UNL");
```

```
  var header = printFileHeader(headerComments);
95  var package = "";//"package default;";
  var classComments = createClassComment(node.nodeText, "1.0", "MMClass");
97  var title = printJavaClass(0, "public", node.nodeText, classComments);
  var attributes = printAttributes(node, connector, dic, VAR_TYPE_BY_INFERECE);
99  var functions = printFunctions(node, dic, VAR_TYPE_BY_INFERECE);
  var constructor = printConstructor(1, "public", node.nodeText, null);
101
  result = header + "\n";
103 result = result + package + "\n\n";
  result = result + title + " {" + "\n\n";
105   result = result + attributes + "\n";
    result =  result + constructor + "\n\n";
107   result = result + functions + "\n";
  result = result + "}";
109
  return result;
111 }
```

D

Cenários recolhidos

Listagem D.1: Cenario 1

```
Feature: Vessel Details
In order to get information about a vessel
As a coast guard
I want to be able to select a vessel and see its details

Scenario: class A vessel
  Given vessel 'Sea Lion' with details:
    | MMSI | 245000000 |
    | Class  |A |
    | Type | Cargo |
    | Status | Underway using engine |
    | Position  | 51.99N, 4.05E |
    | Heading | 290 |
    | Speed | 13.1 |
  When 'Sea Lion' sends a position report
    And 'Sea Lion' sends a voyage report
    And I select vessel 'Sea Lion' on the map
  Then I should see all details of vessel 'Sea Lion'
```

Listagem D.2: Cenario 2

```
Feature: Map View
In order to asses the situation in my area
As a coast guard
I want to see the location of each vessel marked on a map
Scenario: show vessel inside map area
  Given vessel 'Seal' at position '52.01N, 3.99E'
```

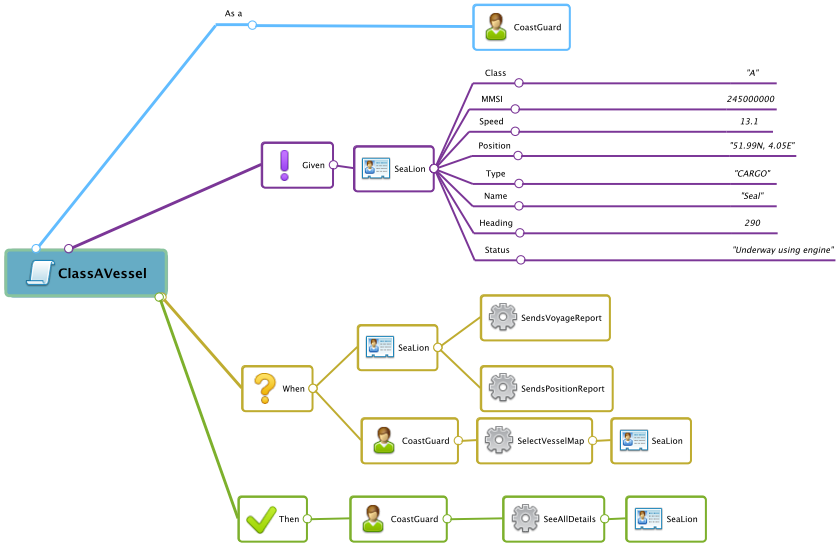


Figura D.1: Cenario 1

```
  When I view the map area between '52.10N, 3.90E' and '51.90N, 4.10E'
8 Then I should see vessel 'Seal' at position '52.01N, 3.99E'
```

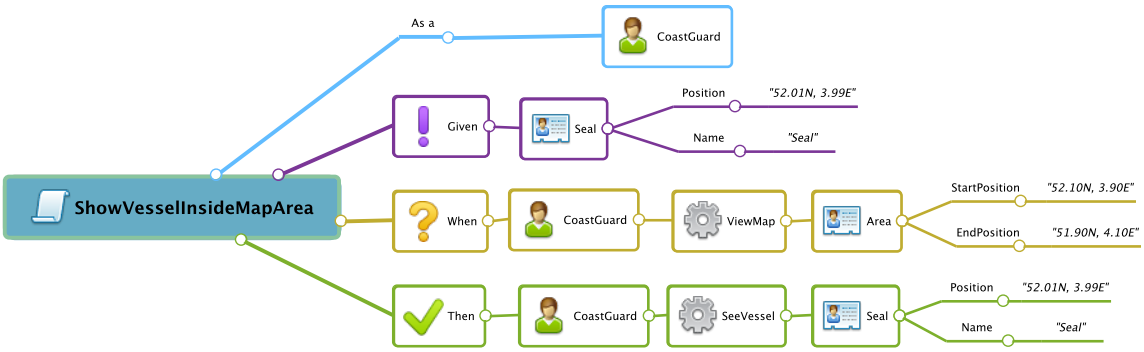


Figura D.2: Cenario 2

Listagem D.3: Cenario 3

```
  Feature: Map View
2 In order to asses the situation in my area
  As a coast guard
4 I want to see the location of each vessel marked on a map
  Scenario: vessels outside the map area should not be visible
6   Given vessel 'Seagull' at position '51.97N, 4.12E'
    When I view the map area between '52.10N, 3.90E' and '51.90N, 4.10E'
8   Then I should not see vessel 'Seagull'
```

Listagem D.4: Cenario 4

```
  Feature: Vessel Speed
2   In order to gauge the speed of vessels
```

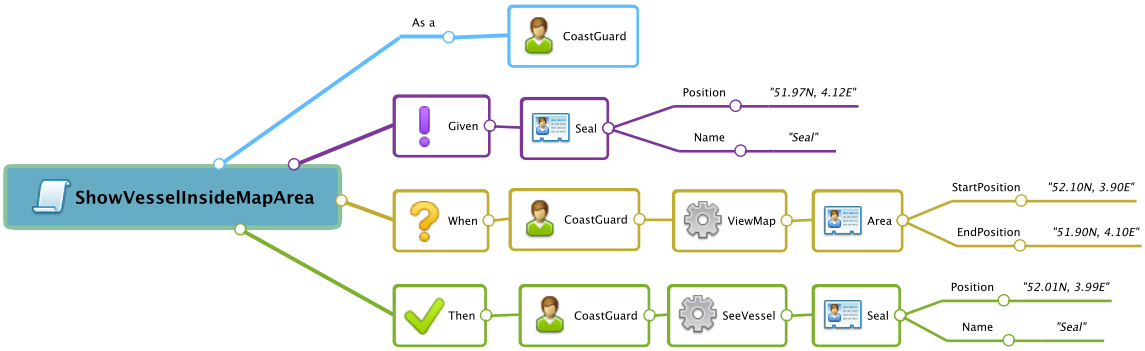


Figura D.3: Cenario 3

```
  As a coast guard
  I want to see a line behind each marked vessel indicating speed
Scenario: regular movement
Given vessels with speeds:
  | name      | speed |
  | Sea Lion  | 10.0  |
  | Seagull   | 20.0  |
  | Seal      | 30.0  |
When I view the map
Then I should see speed lines:
  | name     | relative length |
  | Sea Lion  | 1.0             |
  | Seagull   | 2.0             |
  | Seal     | 3.0             |
```

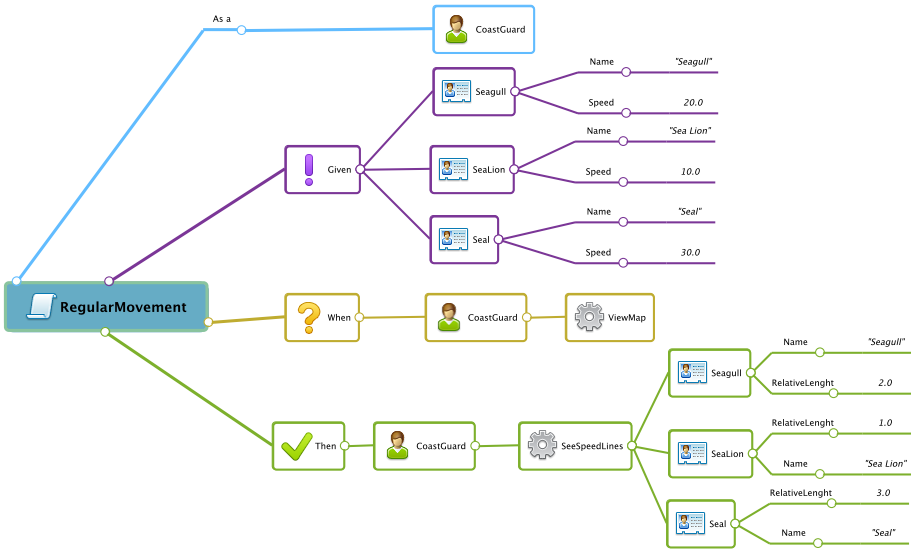


Figura D.4: Cenario 4

Listagem D.5: Cenario 5

```
Feature: Vessel Speed
  In order to gauge the speed of vessels
  As a coast guard
  I want to see a line behind each marked vessel indicating speed
Scenario: slow vessels have a minimum line length
Given vessels with speeds:
  | name       | speed |
  | Sea Lion   | 1.0   |
  | Seagull    | 5.0   |
  | Seal       | 10.0  |
When I view the map
Then I should see speed lines:
  | name      | relative length |
  | Sea Lion  |   1.0           |
  | Seagull   |   1.0           |
  | Seal      |   1.0           |
```

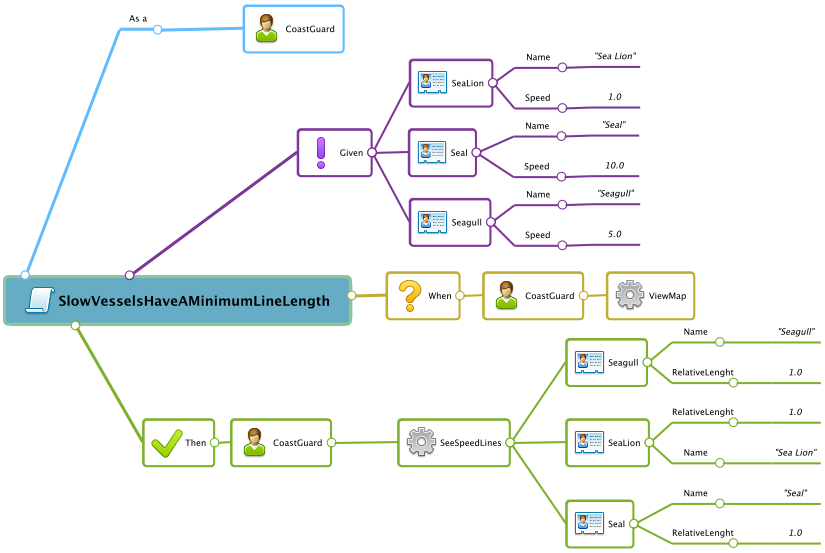


Figura D.5: Cenario 5

Listagem D.6: Cenario 6

```
Feature: Vessel Speed
  In order to gauge the speed of vessels
  As a coast guard
  I want to see a line behind each marked vessel indicating speed
Scenario: fast vessels have a maximum line length
Given vessels with speeds:
  | name      | speed |
  | Sea Lion  | 30.0  |
  | Seagull   | 50.0  |
When I view the map
Then I should see speed lines:
```

```
  | name | relative length |
  | SeaLion | 1.0 |
  | Seagull | 1.0 |
```

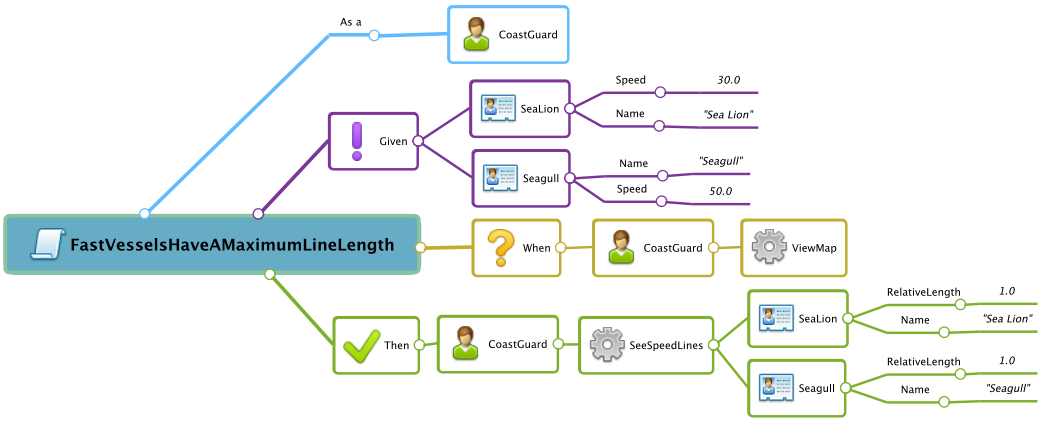


Figura D.6: Cenario 6

Listagem D.7: Cenario 7

```
Scenario: stationary vessels have no lines
Given vessels with speeds:
  | name      | speed |
  | Sea Lion  | 0.0   |
  | Seagull   | 0.99  |
When I view the map
Then I should see no speed lines
```

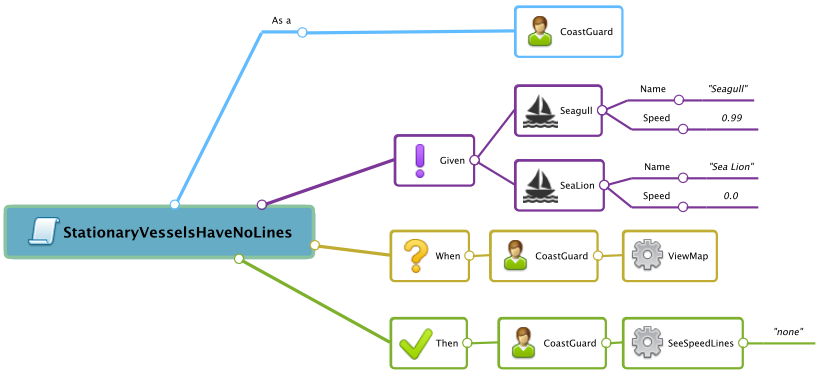


Figura D.7: Cenario 7

Listagem D.8: Cenario 8

```
Scenario: show vessel inside map area

  Given class 'A' vessel 'Seal' at position '52.01N, 3.99E'
  And class 'B' vessel 'Seagull' at position '52.0N, 4.0E'

```

```
  When I see the map area between '52.10N, 3.90E' and '51.90N, 4.10E'

  Then I should see a vessel at position '52.01N, 3.99E'
  And I should see a vessel at position '52.0N, 4.0E'
```

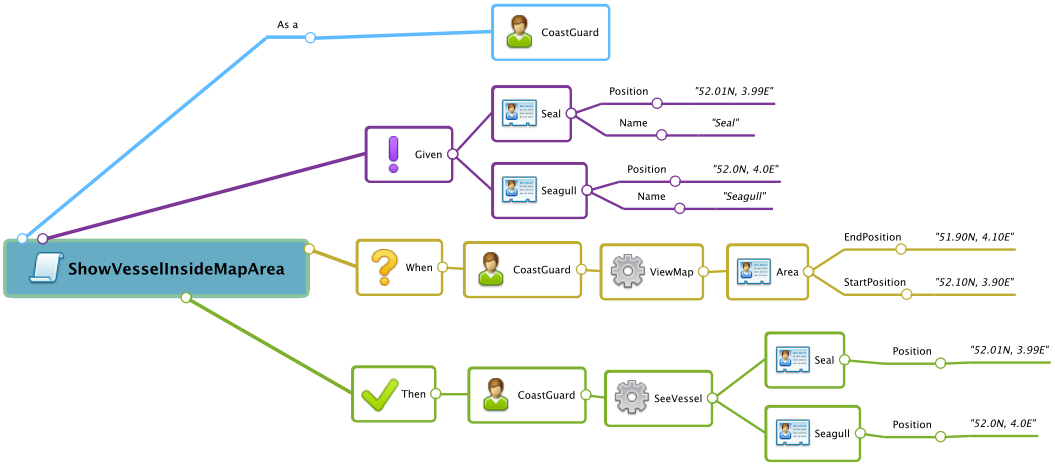


Figura D.8: Cenario 8

Listagem D.9: Cenario 9

```
Vessel Compliance
  In order to see how reliable vessel information is
  As a coast guard
  I want to see which class A vessels do not comply with the AIS protocol
Scenario: dynamic information of moving, non-anchored vessels not changing
    course
Given non-anchored class 'A' vessels with dynamic information:
  | name       | speed |
  | Sea Lion   |1.0    |
  | Seal       |1.0    |
  | Seagull    |14.1   |
  | Seahorse   |14.1   |
  | Sea Otter |23.1   |
  | Seahawk    |23.1   |
When these vessels send a position report And send another position report
    after:
  | name        | interval |
  | Sea Lion    | 10.0     |
  | Seal      | 10.1     |
  | Seagull     | 6.0      |
  | Seahorse    | 6.1      |
  | Sea Otter   | 2.0      |
  | Seahawk     | 2.1      |
Then the compliance of the vessels should be marked as:
  | name       | compliant |
  | Sea Lion   | yes       |
  | Seal       | no        |
```

```
    | Seagull    | yes       |
27  | Seahorse   | no        |
    | Sea Otter  | yes       |
29  | Seahawk    | no        |
```

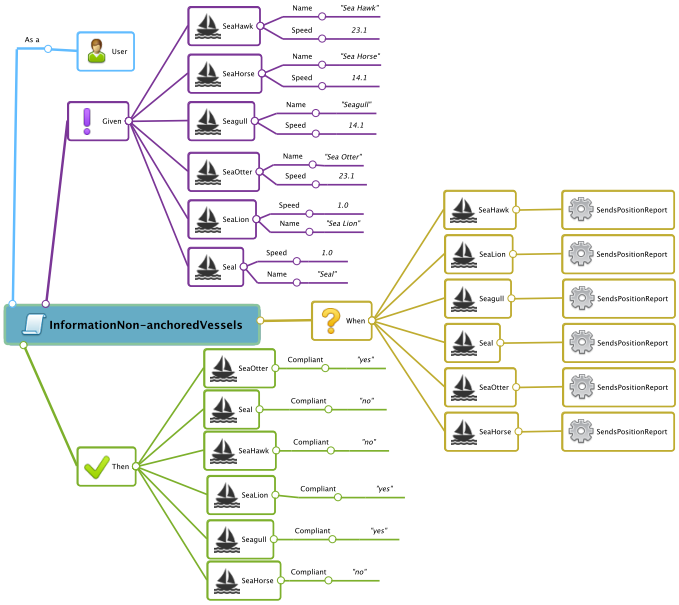


Figura D.9: Cenario 9

Listagem D.10: Cenario 10

```
1   Feature: Area Speed Definition
    In order to see if vessels are in an area with a speed restriction
3   As a coast guard
    I want to define an area with a speed restriction
5   Scenario: define an area

7     Given I see the map area between '52.10N, 3.90E' and '51.90N, 4.10E'
      When I define an area with a maximum speed of '19.0' knots and coords:
9       | 52.00N, 3.97E |
        | 52.02N, 3.98E |
11      | 52.00N, 3.99E |
      And I refresh the page
13    And I see the map area between '52.10N, 3.90E' and '51.90N, 4.10E'
      Then I should see the area with restrictions marked on the map
```

Listagem D.11: Cenario 11

```
    Feature: Area Speed Compliance
2   In order to see vessels that do not comply with area rules
    As a coast guard
4   I want to see which vessels move too fast in an area with speed limits
    Scenario: vessel inside area, moving too fast
6     Given class 'A' vessel 'Seal' at position '52.01N, 3.98E'
```

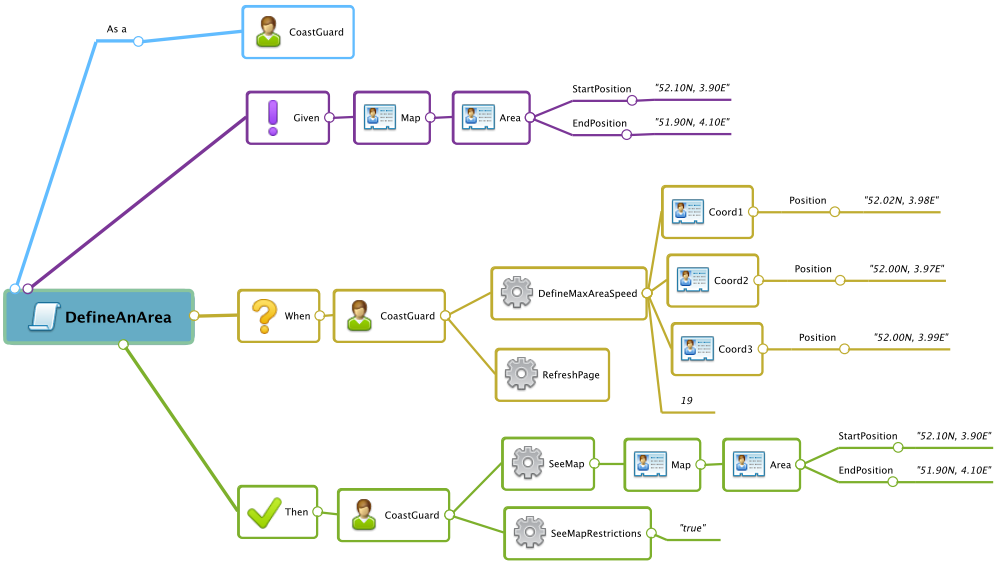


Figura D.10: Cenario 10

```
    And vessel 'Seal' has a speed of '25.0' knots
8   And an area with a maximum speed of '15.0' knots and coords:
      | 52.00N, 3.97E |
10    | 52.02N, 3.98E |
      | 52.00N, 3.99E |
12  When I see the map area between '52.10N, 3.90E' and '51.90N, 4.10E'
    Then vessel 'Seal' should be shown as non-compliant
```

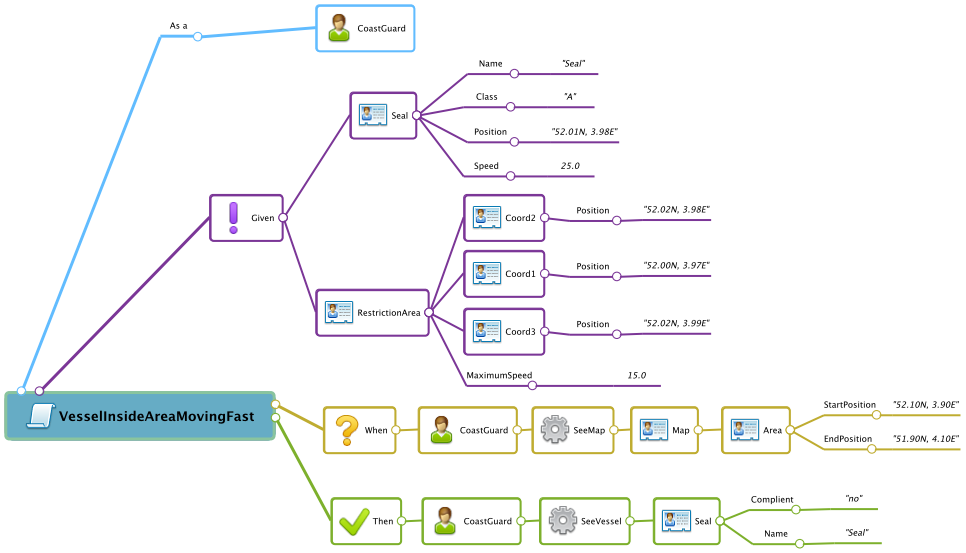


Figura D.11: Cenario 11

Listagem D.12: Cenario 12

```
1  Given a customer:
```

```
    |CustomerCode|CustomerName|Country|Currency|
    |6538764 |John Smith |USA |USD |
And a couple of portfolios :
    |CustomerCode|PortfolioCode|PortfolioType |
    |6538764 |FIFO_1 |FIFO |
    |6538764 |AVG_1 |Average priced|
And a security:
    |SecurityID|SecurityName|Currency|Decimals|
    |NOK |Nokia |USD |2 |
And some purchase lots in both portfolios :
    |TransactionType|SecurityID|Amount|Portfolio |
    |Purchase |NOK |172.12|6538764/FIFO_1|
    |Sale |NOK |83.47 |6538764/FIFO_1|
    |Purchase |NOK |100.00|6538764/AVG_1 |
When the current position for customer '6538764' is requested
Then the system should display :
    |CustomerCode|PortfolioCode|SecurityID|Amount|
    |6538764 |FIFO_1 |NOK |88.65 |
    |6538764 |AVG_1 |NOK |100.00|
```

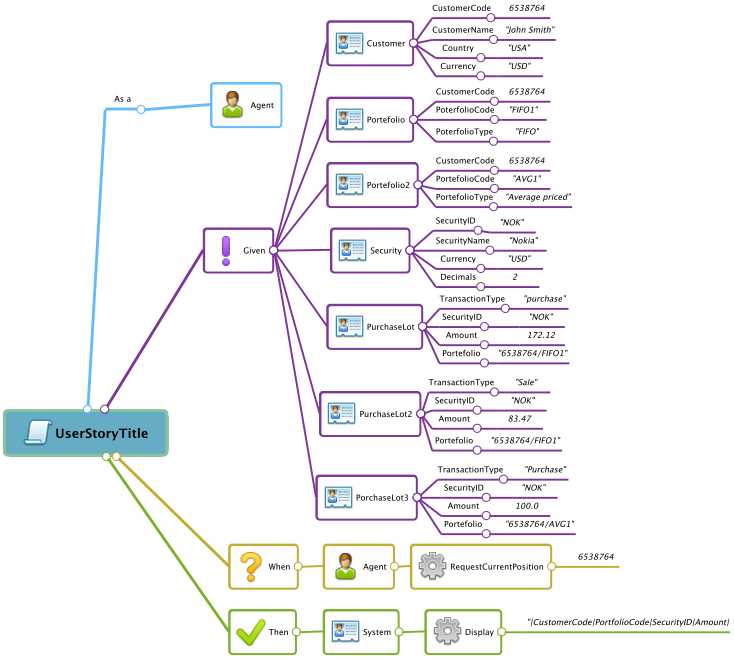


Figura D.12: Cenario 12

Listagem D.13: Cenario 15

```
Feature: Showtime Descriptions
  So that I can find movies that fit my schedule
  As a movie goer
  I want to see accurate and concise showtimes
Scenario: Show minutes for times not ending with 00
  Given a movie
```

```
  When I set the showtime to 2007-10-10 at 2:15pm
  Then the showtime description should be 'October 10, 2007 (2:15pm)'
```

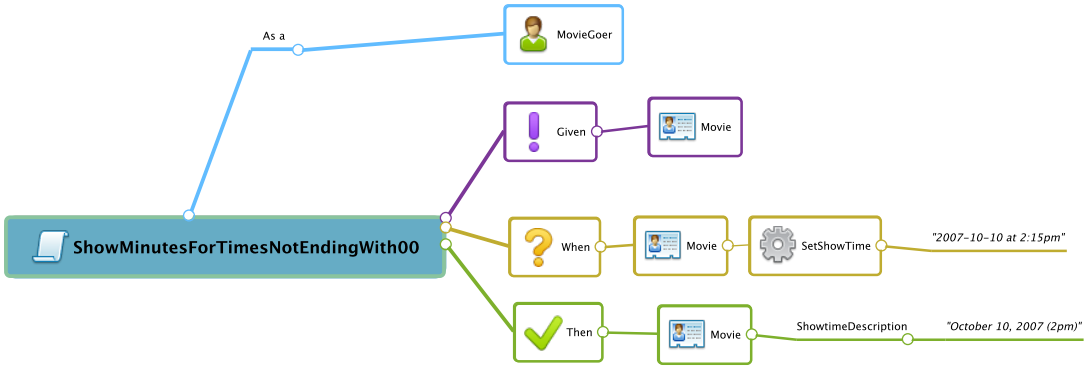


Figura D.13: Cenario 15

Listagem D.14: Cenario 16

```
Feature: Showtime Descriptions
  So that I can find movies that fit my schedule
  As a movie goer
  I want to see accurate and concise showtimes
Scenario: Hide minutes for times ending with 00
  Given a movie
  When I set the showtime to 2007-10-10 at 2:00pm
  Then the showtime description should be 'October 10, 2007 (2pm)'
```

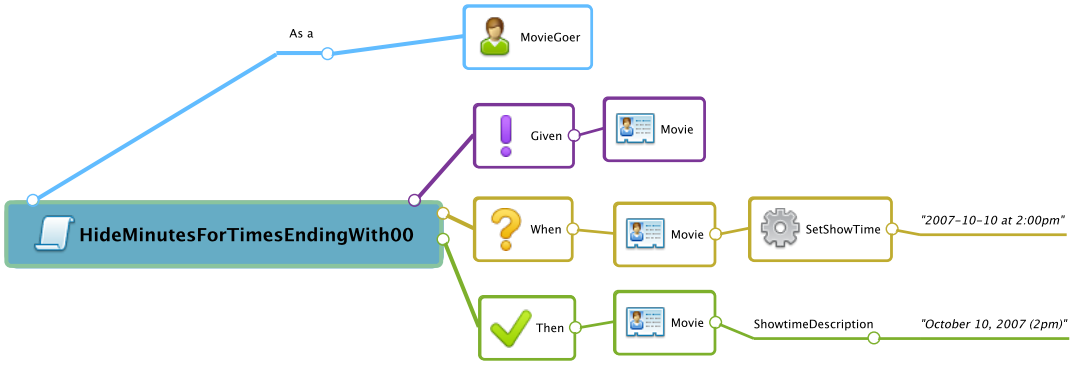


Figura D.14: Cenario 16

Listagem D.15: Cenario 17

```
Feature: Browse Movies
  So that I quickly can find movies of interest
  As a movie goer
  I want to browse movies by genres
  Scenario: Add movie to genre
    Given a genre named Comedy
```

```
    When I create a movie Caddyshack in the Comedy genre
    Then Caddyshack should be in the Comedy genre
```

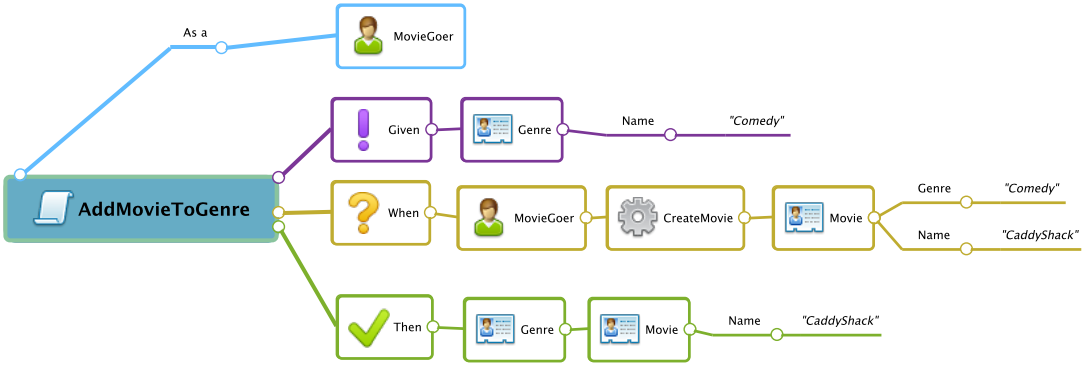


Figura D.15: Cenario 17

Listagem D.16: Cenario 18

```
Scenario:  trader is not alerted below threshold

Given a stock of symbol STK1 and a threshold of 10.0
When the stock is traded at 5.0
Then the alert status should be OFF
```

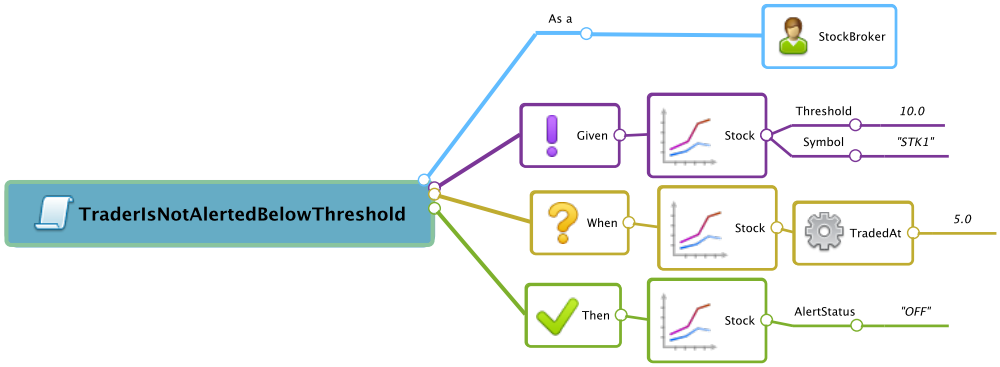


Figura D.16: Cenario 18

Listagem D.17: Cenario 19

```
Scenario: Add two numbers
Given I have entered 50 into the calculator
And I have entered 70 into the calculator
When I press Add
The the result should be 120 on the screen
```

Listagem D.18: Cenario 20

```
Feature: Adding
```

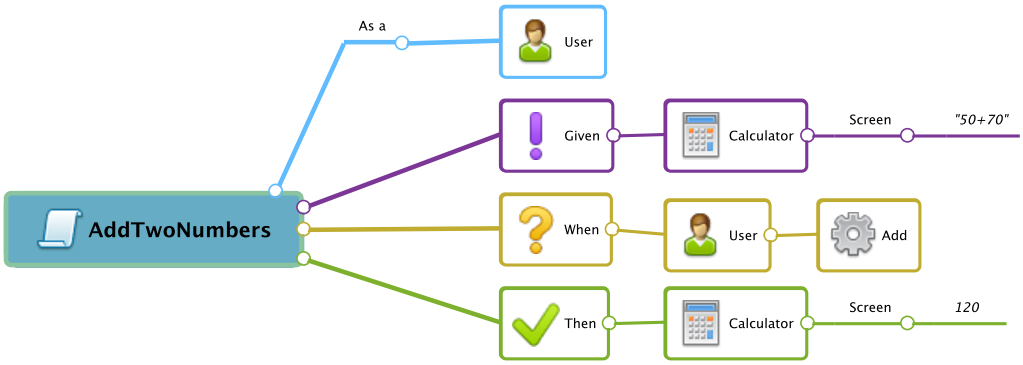


Figura D.17: Cenario 19

```
  Scenario: Add two numbers
3 Given the input '2+2'
  When the calculator is run
5 Then the output should be '4'
```

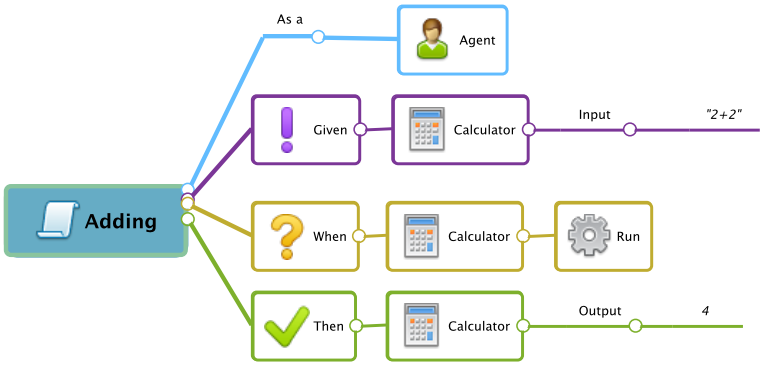


Figura D.18: Cenario 20

Listagem D.19: Cenario 21

```
1 Account Holder withdraws cash

3 As an Account Holder
  I want to withdraw cash from an ATM
5 So that I can get money when the bank is closed
  Scenario 1: Account has sufficient funds
7 Given the account balance is \$100
   And the card is valid
9  And the machine contains enough money
  When the Account Holder requests \$20
11 Then the ATM should dispense \$20
   And the account balance should be \$80
13  And the card should be returned
```

Listagem D.20: Cenario 22

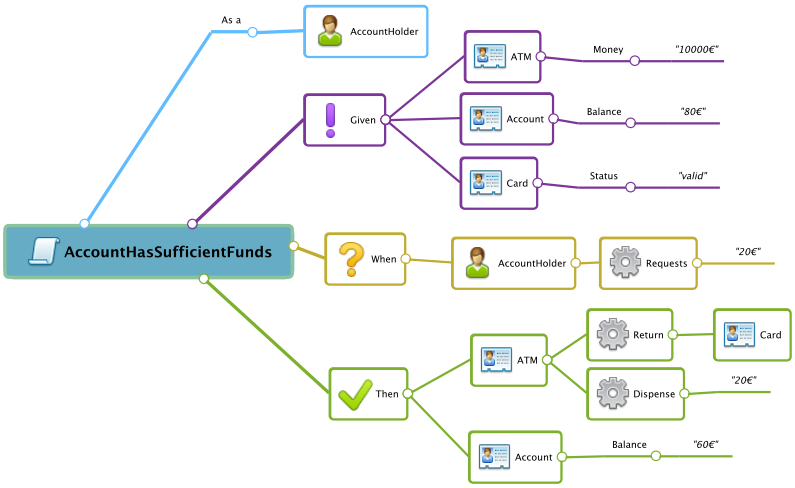


Figura D.19: Cenario 21

```
Account Holder withdraws cash

As an Account Holder
I want to withdraw cash from an ATM
So that I can get money when the bank is closed
Scenario 2: Account has insufficient funds
Given the account balance is \$10
 And the card is valid
 And the machine contains enough money
When the Account Holder requests \$20
Then the ATM should not dispense any money
 And the ATM should say there are insufficient funds
 And the account balance should be \$20
 And the card should be returned
```

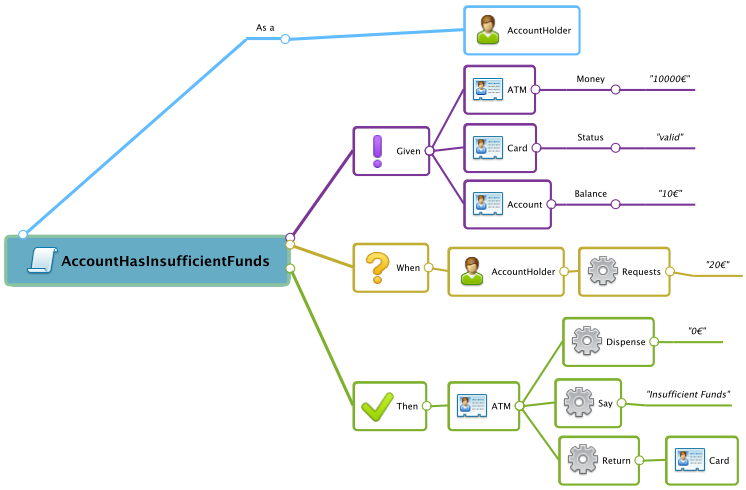


Figura D.20: Cenario 22

Listagem D.21: Cenario 23

```
Account Holder withdraws cash

As an Account Holder
I want to withdraw cash from an ATM
So that I can get money when the bank is closed
Scenario 3: Card has been disabled
Given the card is disabled
 When the Account Holder requests \$20
 Then the ATM should retain the card
  And the ATM should say the card has been retained
```

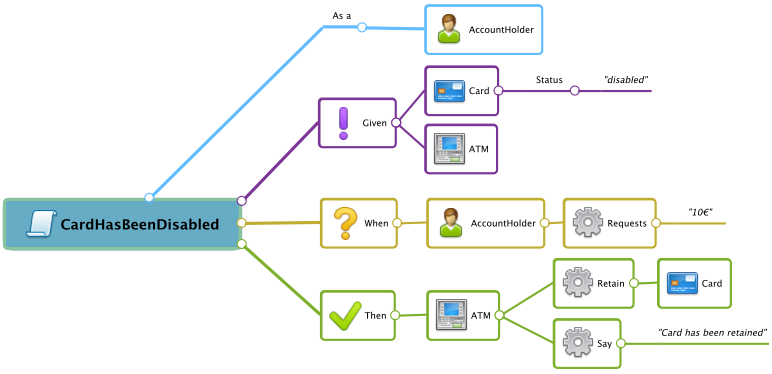


Figura D.21: Cenario 23

Listagem D.22: Cenario 24

```
Returns go to stock

In order to keep track of stock
As a store owner
I want to add items back to stock when they're returned
Scenario 2: Replaced items should be returned to stock
Given that a customer buys a blue garment
And I have two blue garments in stock
And three black garments in stock.
When he returns the garment for a replacement in black,
Then I should have three blue garments in stock
And two black garments in stock
```

Listagem D.23: Cenario 25

```
Scenario: User with valid credentials

Given an unauthenticated user
When the user tries to access a restricted asset
Then they should be directed to a login page

(CEN_XX)
```

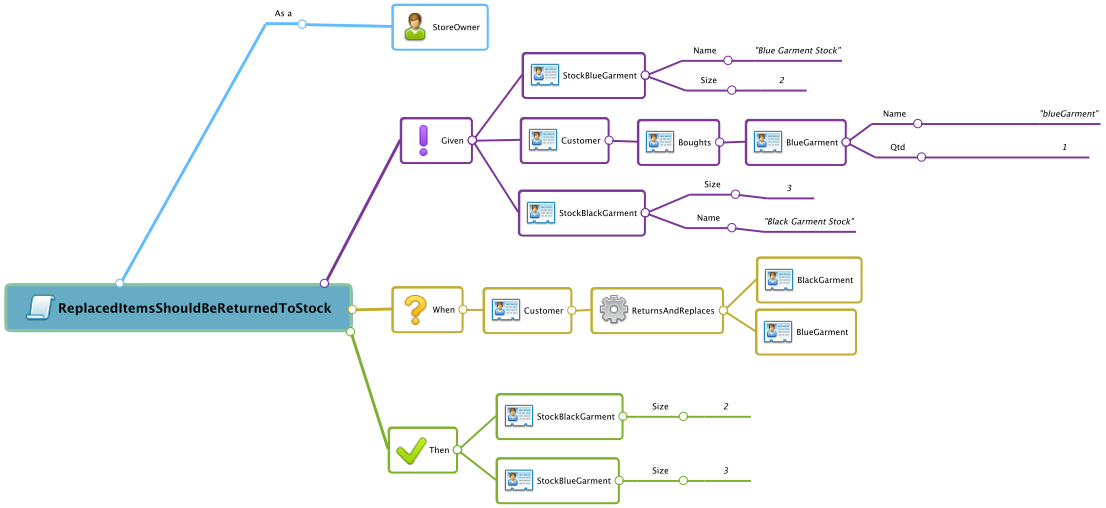


Figura D.22: Cenario 24

```
Given an unauthenticated user
When the user submits valid credentials
Then they should be redirected back to the restricted content
```

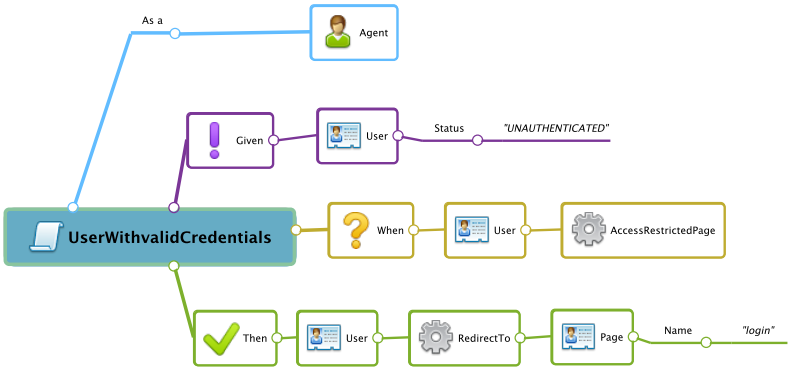


Figura D.23: Cenario 25

Listagem D.24: Cenario 27

```
Scenario: Check inbox
  Given a User 'Dave' with password 'password'
  And a User 'Sue' with password 'secret'
  And an email to 'Dave' from 'Sue'
  When I sign in as 'Dave' with password 'password'
  Then I should see 1 email from 'Sue' in my inbox
```

Listagem D.25: Cenario 28

```
Feature: Display balance
Scenario: User checks the balance of an account in credit

Given my account has been credited with $100
```

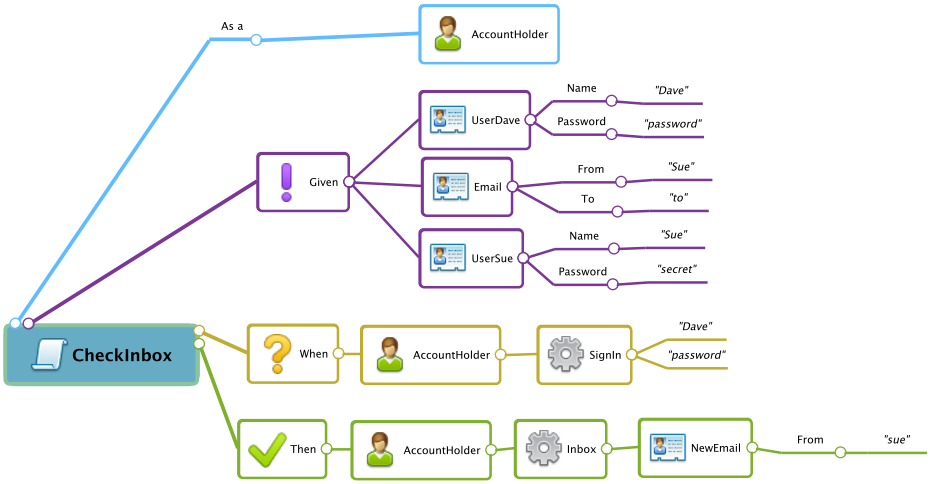


Figura D.24: Cenario 27

```
When I check my balance
Then I should see that my balance is $100
```

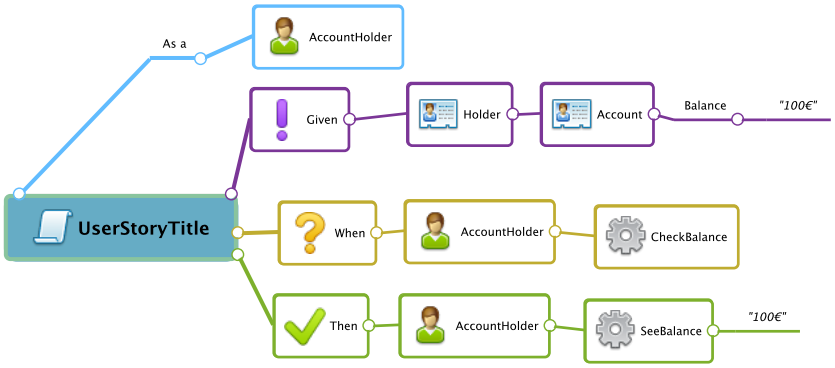


Figura D.25: Cenario 28

Listagem D.26: Cenario 29

```
Scenario: Attempt withdrawal using stolen card
  Given I have $100 in my account
  Given my card is invalid
  When I request $50
  Then my card should not be returned
  Then I should be told to contact the bank
```

Listagem D.27: Cenario 30

```
Scenario: Change PIN successfully
  Given I have been issued a new card
  And I insert the card, entering the correct PIN
  When I choose 'Change PIN' from the menu
  And I change the PIN to 9876
```

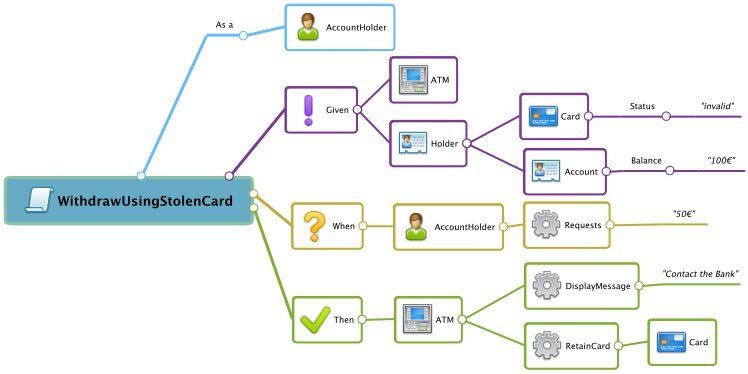


Figura D.26: Cenario 29

```
6   Then the system should remember my PIN is now 9876
```

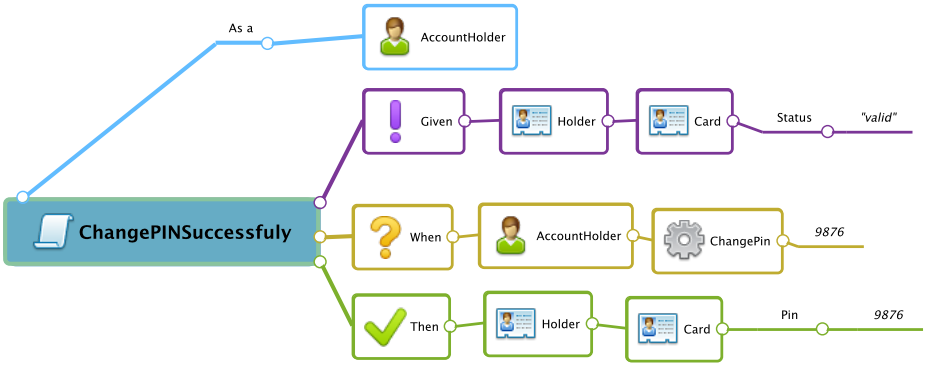


Figura D.27: Cenario 30

Listagem D.28: Cenario 31

```
Scenario: Try to change PIN to the same as before
2   Given I have been issued a new card
    And I insert the card, entering the correct PIN
4   When I choose 'Change PIN' from the menu
    And I try to change the PIN to the original PIN number
6   Then I should see a warning message
    And the system should not have changed my PIN
```

Listagem D.29: Cenario 32

```
1 Scenario: Withdraw fixed amount of $50
    Given I have $500 in my account
3   When I choose to withdraw the fixed amount of $50
    Then I should receive $50 cash
5   And the balance of my account should be $450
```

Listagem D.30: Cenario 33

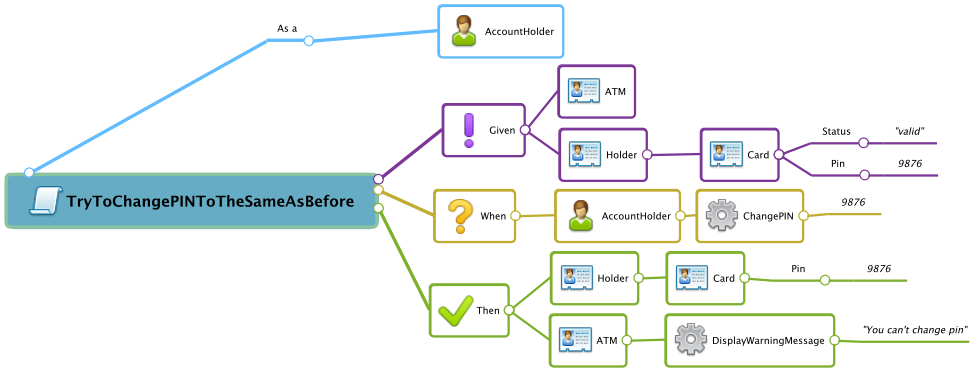


Figura D.28: Cenario 31

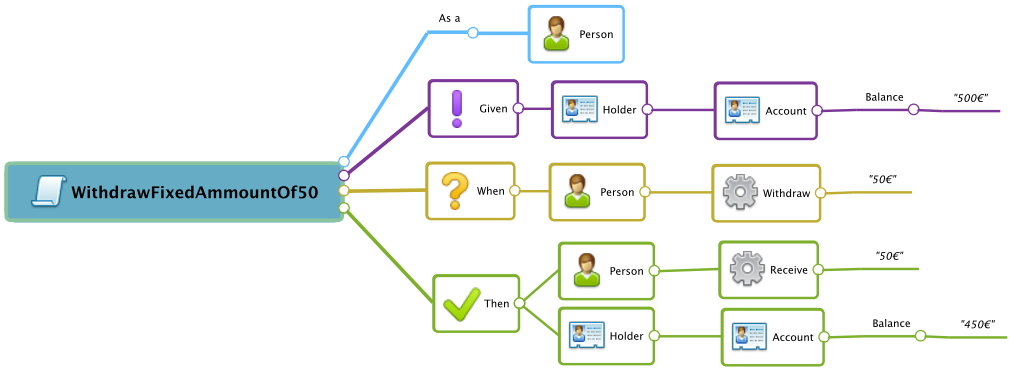


Figura D.29: Cenario 32

```
Scenario: Withdraw fixed amount of $100
  Given I have $500 in my account
  When I choose to withdraw the fixed amount of $100
  Then I should receive $100 cash
  And the balance of my account should be $400
```

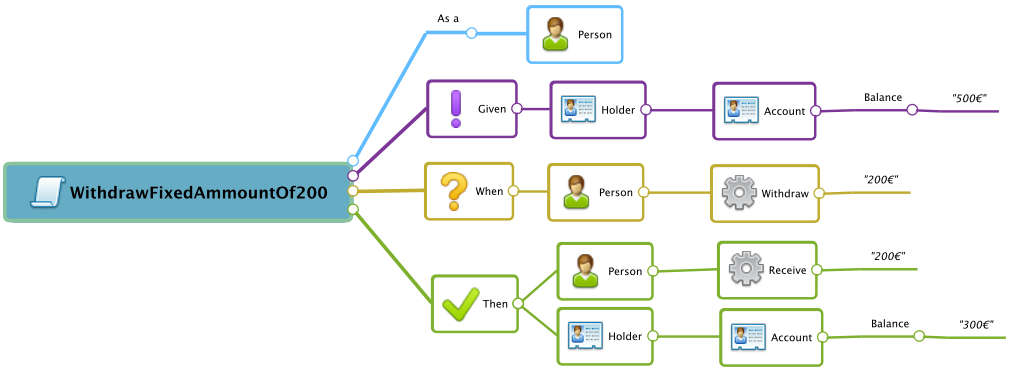


Figura D.30: Cenario 33

Listagem D.31: Cenario 34

```
Scenario: Withdraw fixed amount of $200
  Given I have $500 in my account
  When I choose to withdraw the fixed amount of $200
  Then I should receive $200 cash
  And the balance of my account should be $300
```

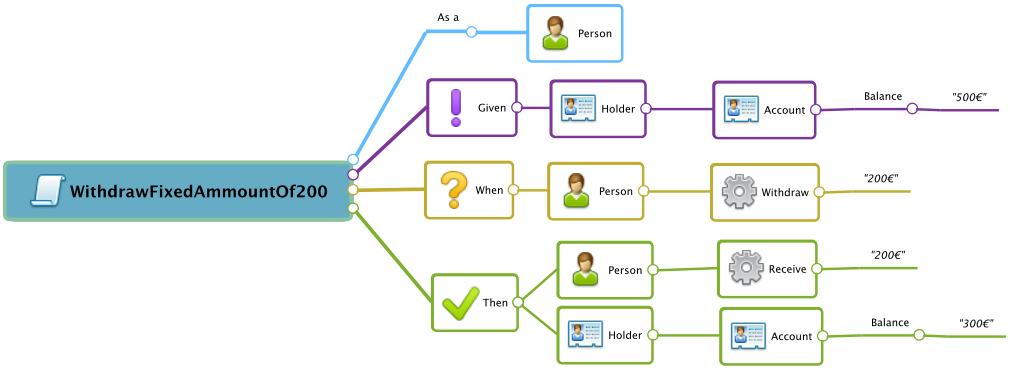


Figura D.31: Cenario 34

Listagem D.32: Cenario 35

```
Feature:
  Scenario:
  Given a board like this:
    | |1|2|3|
    |1| | | |
    |2| | | |
```

```
    |3| | | |
  When player x plays in row 2, column 1
  Then the board should look like this:
    | |1|2|3|
    |1| | | |
    |2|x| | |
    |3| | | |
```

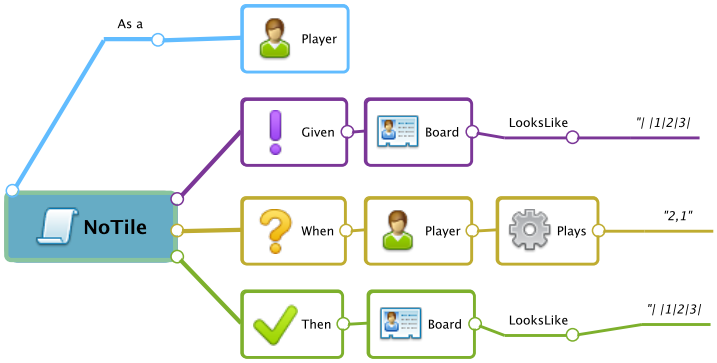


Figura D.32: Cenario 35

Listagem D.33: Cenario 36

```
Scenario: Renaming headers
Given I am logged in as a buyer
When I search for available cars
Then I should see the following cars:
    | color | model   |
    | rust  | Camaro  |
    | blue  | Gremlin |
```

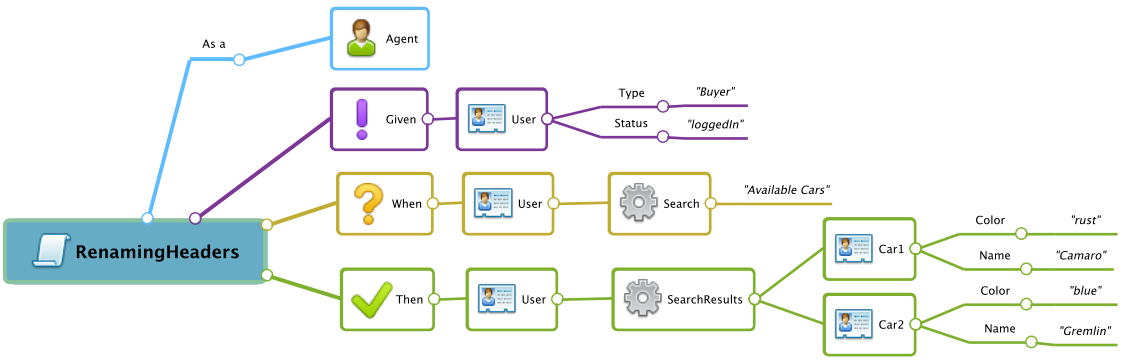


Figura D.33: Cenario 36

Listagem D.34: Cenario 37

```
Scenario: Converting cells
Given I am logged in as a buyer
```

```
When I view warranty options
Then I should see the following options:
    | name     | price |
    | Platinum | $1000 |
    | Gold     | $500  |
    | Silver   | $200  |
```

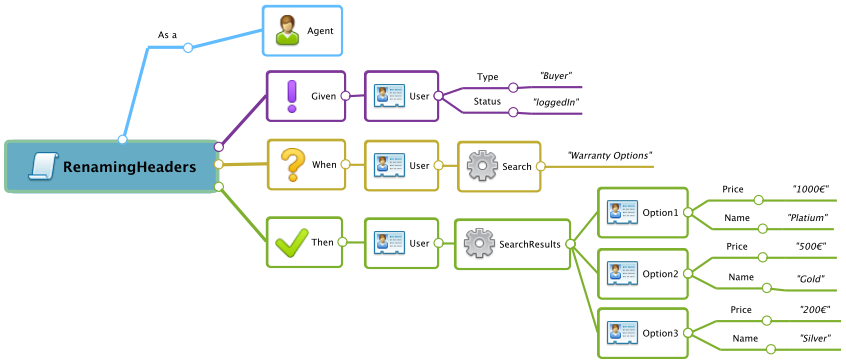


Figura D.34: Cenario 37

Listagem D.35: Cenario 38

```
Feature: Humpty Dumpty
Scenario: Fall
  Given I am on a wall
  When I lose my balance
  Then I should have a great fall
```

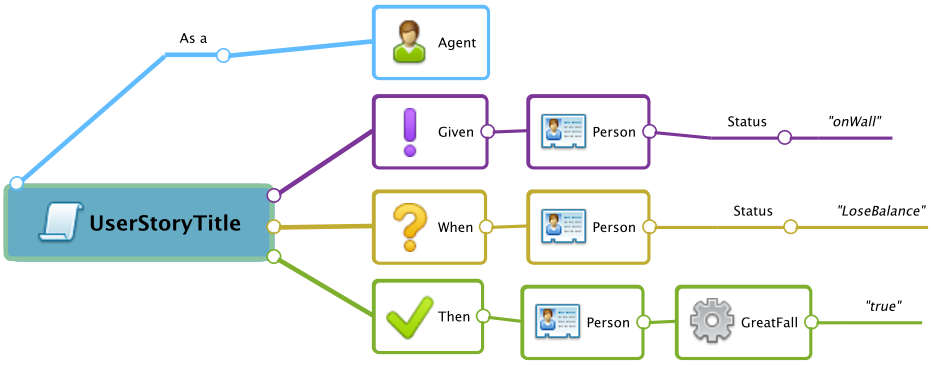


Figura D.35: Cenario 38

Listagem D.36: Cenario 40

```
Feature: Greet user
Scenario: Greet users who are logged in
  Given I am logged in as 'matt'
  When I visit the homepage
  Then I should see 'Hello matt'
```

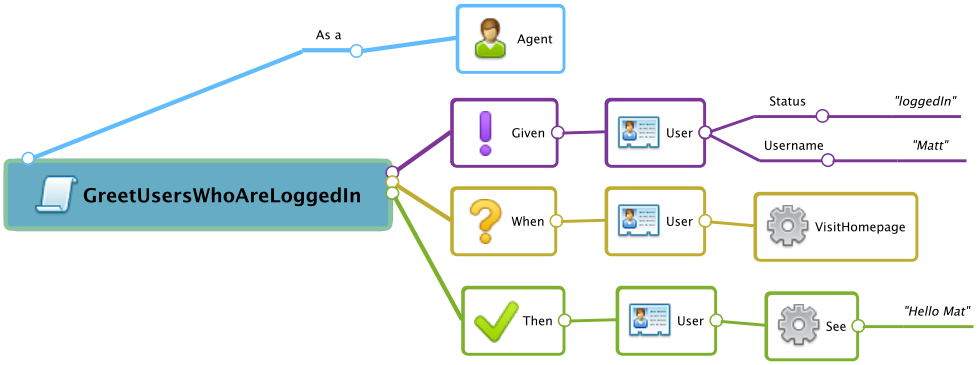


Figura D.36: Cenario 40

Listagem D.37: Cenario 41

```
Feature: Book landing page
Scenario: Related titles
  Given I am on the page for 'Cucumber Recipes'
  When I look for related titles
  Then I should see 'The Cucumber Book'
```

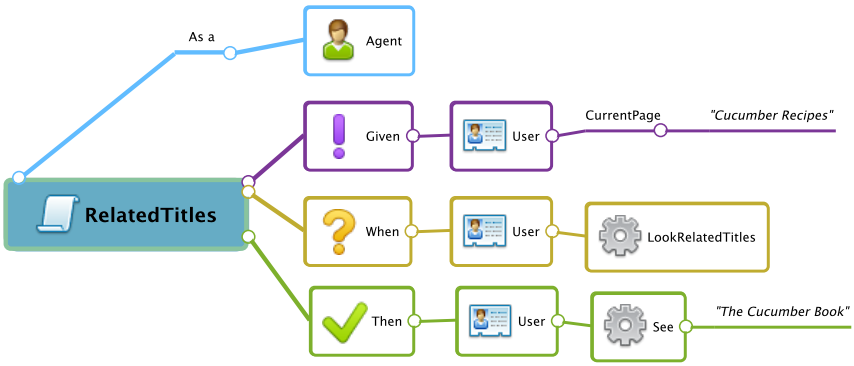


Figura D.37: Cenario 41

Listagem D.38: Cenario 42

```
Feature: Drawing
Scenario: Green circle
  Given a white background
  When I draw a green circle
  Then the result should resemble 'circle.png'
```

Listagem D.39: Cenario 43

```
Feature: Shipping
Scenario: Packing the containers
  Given an order for 20 tons of material
  When I pack 4 shipping containers
  Then the order should be complete
```

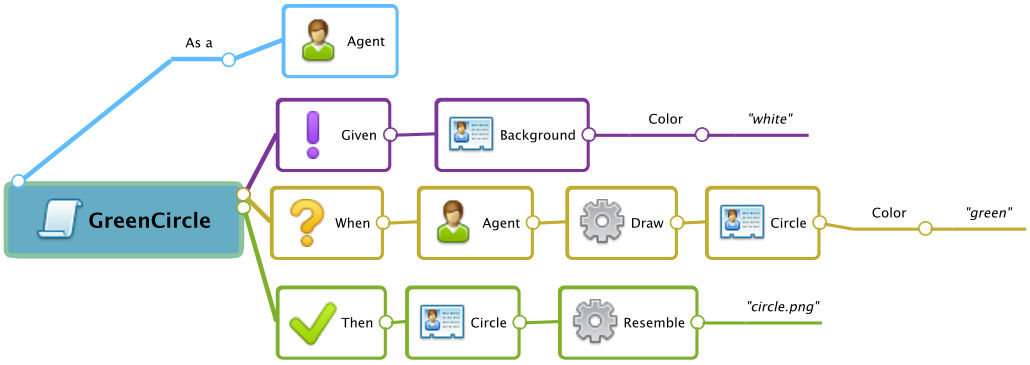


Figura D.38: Cenario 42

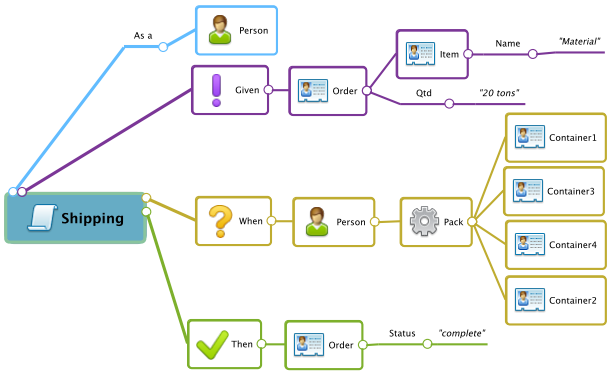


Figura D.39: Cenario 43

Listagem D.40: Cenario 44

```
Feature: Receiving
Scenario: Filling the warehouse
  Given I have received 20 tons of raw material
  When I unload the order into the warehouse
  Then I should have 15% space remaining
```

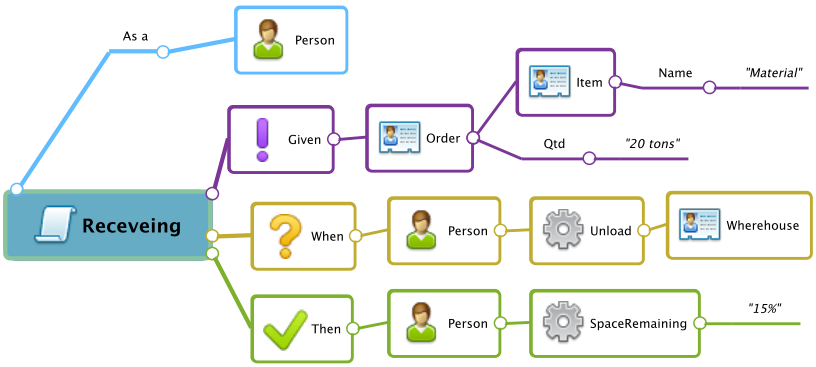


Figura D.40: Cenario 44

Listagem D.41: Cenario 47

```
Scenario: Initially empty log
  Given a log containing: ''''''
  When I append the warning 'Disk space low'
  Then the log should read: '''W Disk space low'''
```

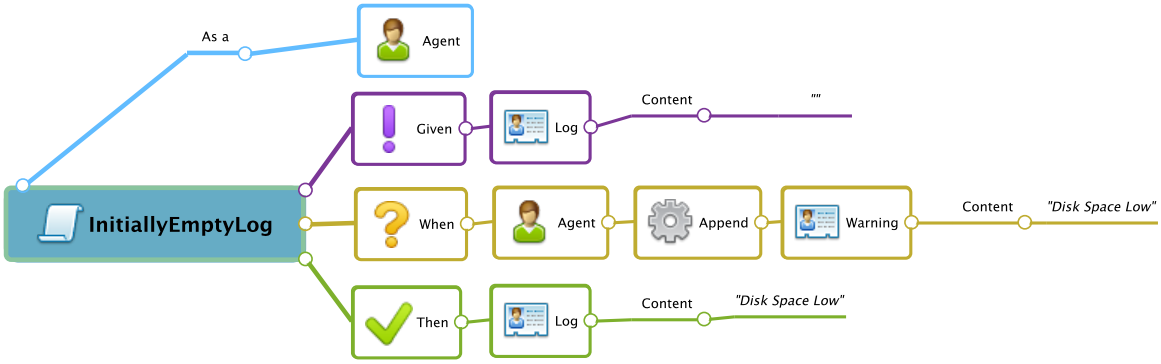


Figura D.41: Cenario 47

Listagem D.42: Cenario 48

```
Feature: Parsing a log
Scenario: Multiple lines
  Given a log containing: '''W Disk space low I Backup complete'''
  When I parse the log
  Then the entries should be:
        | priority    | message          |
```

```
        | warning     | Disk space low  |
        | information | Backup complete |
```

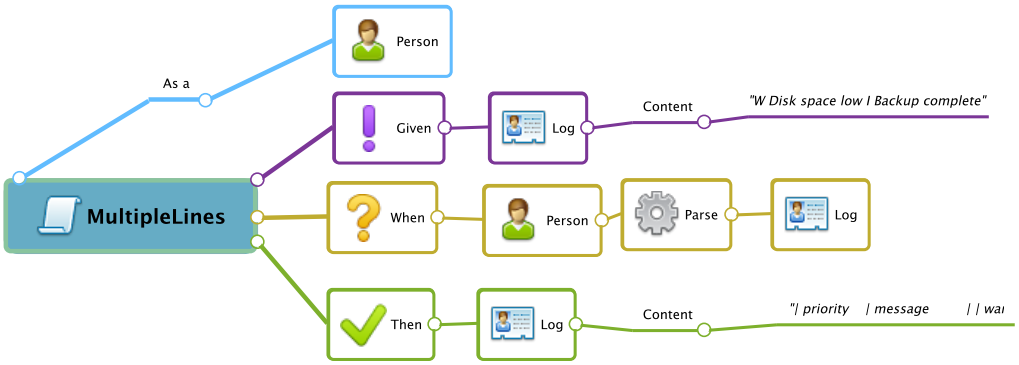


Figura D.42: Cenario 48

Listagem D.43: Cenario 49

```
Feature: Cone of silence
Scenario: Activation
  Given I am writing a book
  When I activate the cone of silence
  Then I should not hear my children for the next hour
```

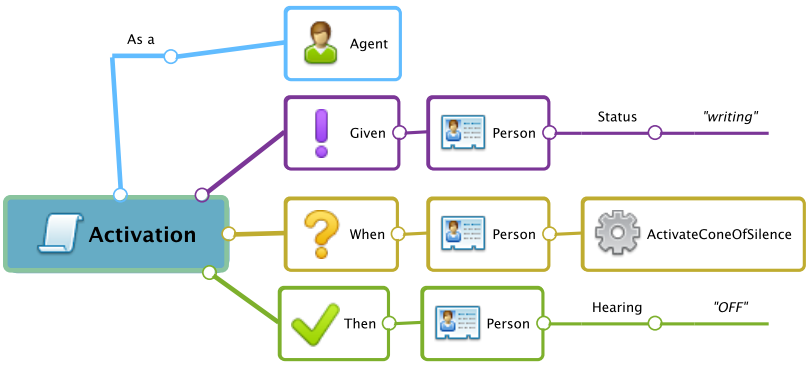


Figura D.43: Cenario 49

Listagem D.44: Cenario 50

```
Feature: Soda machine Scenario: Get soda
  Given I have $2 in my account
  When I wave my magic ring at the machine
  Then I should get a soda
```

Listagem D.45: Cenario 51

```
Feature: Stock broker
Scenario: Buy low, sell high
```

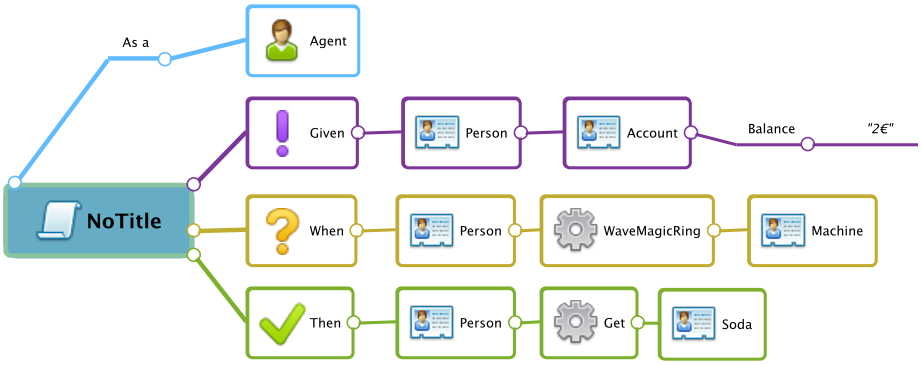


Figura D.44: Cenario 50

```
Given I have 100 shares of GOOG
When I sell all my GOOG shares for $800.00/share
Then I should have $80000.00
```

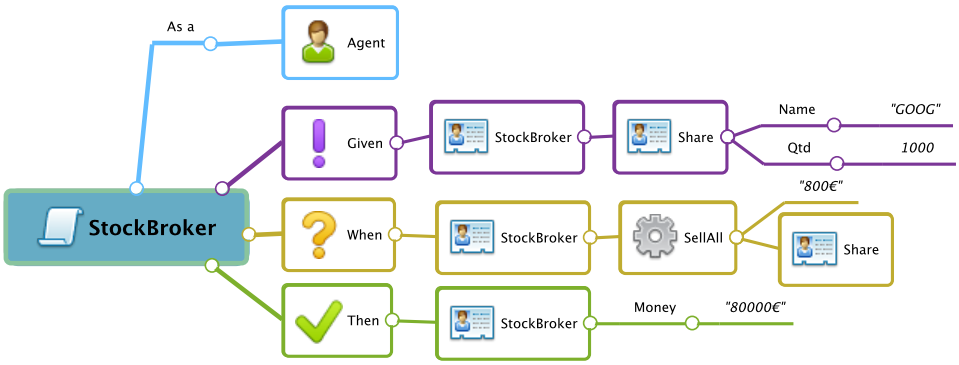


Figura D.45: Cenario 51

Listagem D.46: Cenario 53

```
Story: empty stack
    scenario null is pushed onto empty stack
      given an empty stack
      when null is pushed
      then an exception should be thrown
      then the stack should still be empty
```

Listagem D.47: Cenario 54

```
Story: empty stack
    scenario pop is called on empty stack
      given an empty stack
      when pop is called
      then an exception should be thrown
      then the stack should still be empty
```

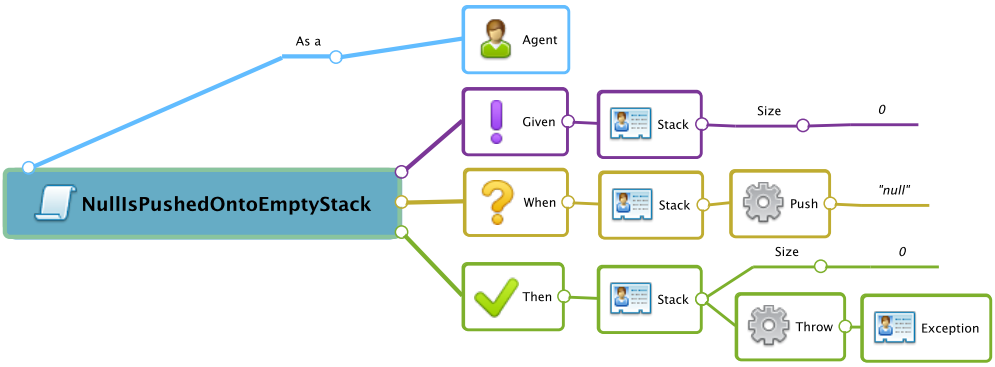


Figura D.46: Cenario 53

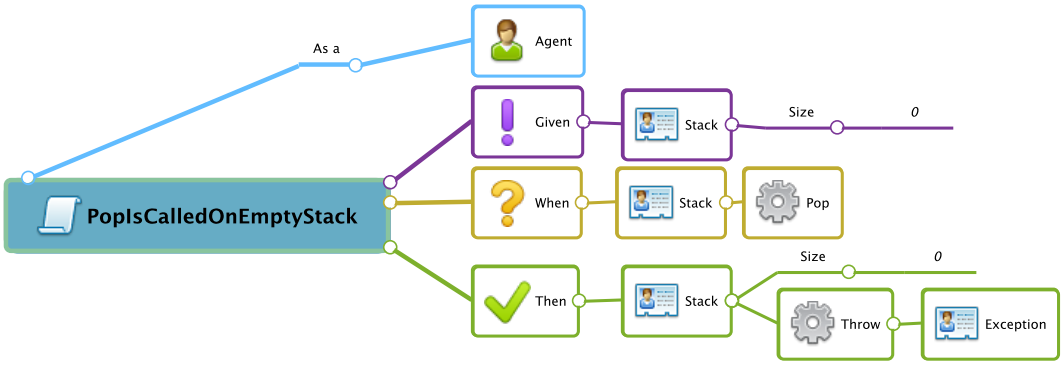


Figura D.47: Cenario 54

Listagem D.48: Cenario 55

```
Story: single value stack
    scenario pop is called on stack with one value
      given an empty stack with one pushed value
      when pop is called
      then that object should be returned
      then the stack should be empty
```

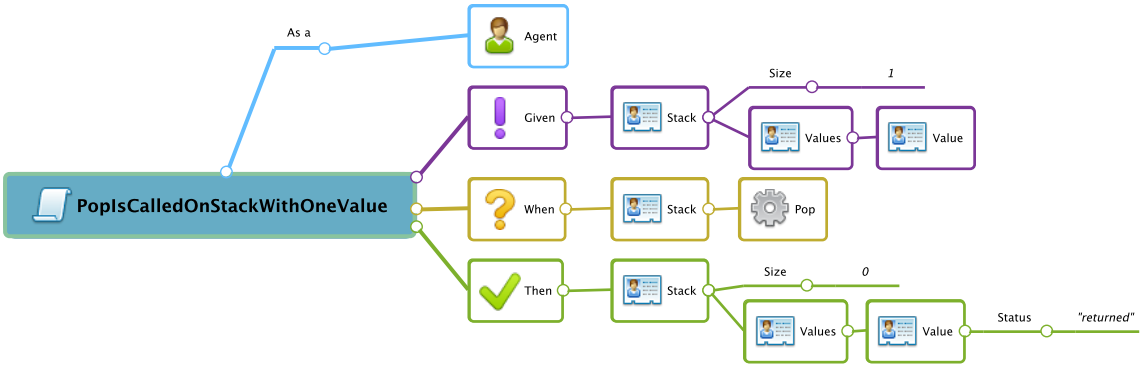


Figura D.48: Cenario 55

Listagem D.49: Cenario 58

```
Feature: ls
  In order to see the directory structure
  As a UNIX user
  I need to be able to list the current directory's contents
  Scenario: List 2 files in a directory
    Given I am in a directory 'test'
    And I have a file named 'foo'
    And I have a file named 'bar'
    When I run 'ls'
    Then I should get:
      '''
      bar
      foo
      '''
```

Listagem D.50: Cenario 60

```
Scenario: Lists with things in them are not empty.
  Given a new list
   when we add an object
   then the list should not be empty.
```

Listagem D.51: Cenario 61

```
Feature: New account
In order to have new ready to use account
```

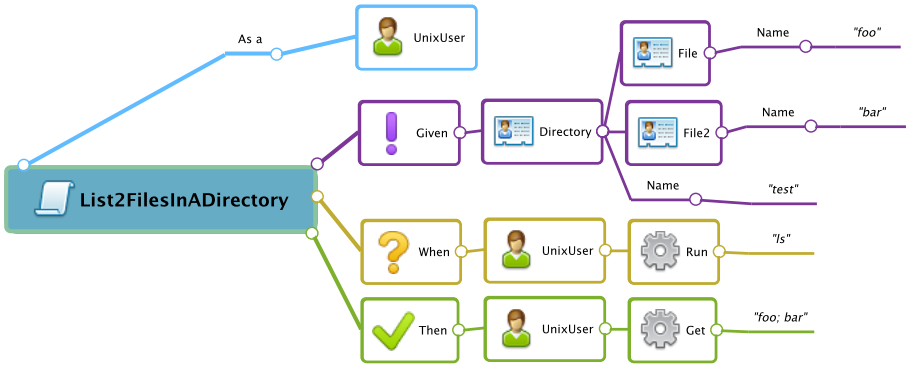


Figura D.49: Cenario 58

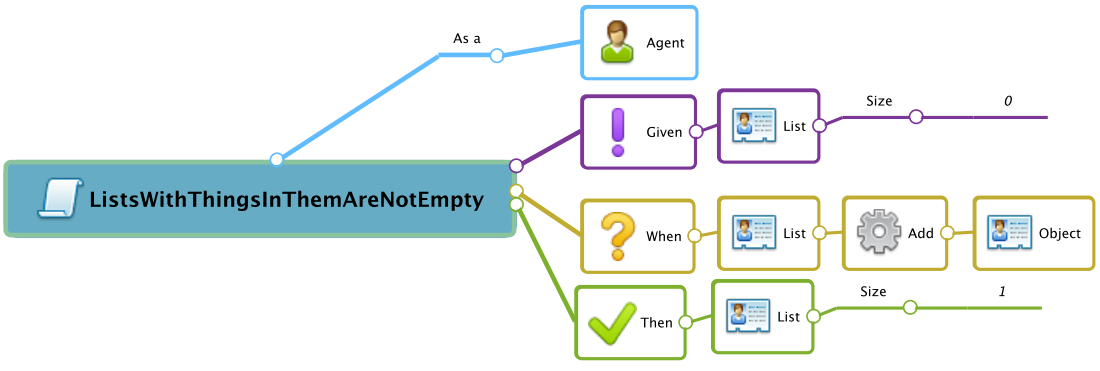


Figura D.50: Cenario 60

```
As a user
I want to get an account with balance 0$
Scenario: Create New Account
  Given I have no account
  When I create an account
  Then I should have an account with balance 0$
```

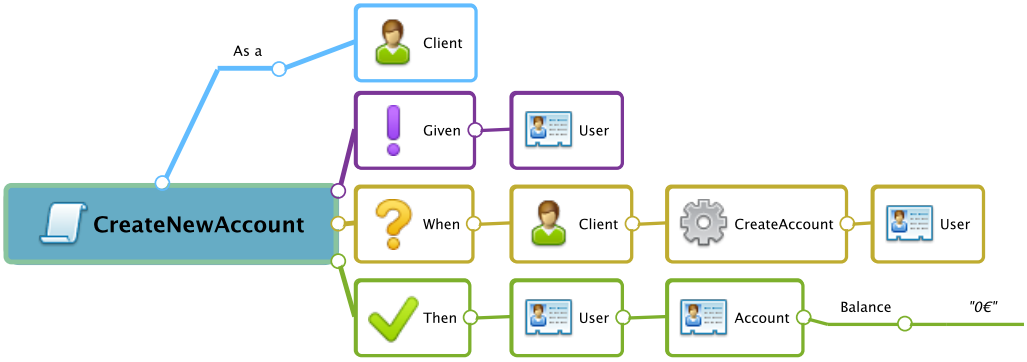


Figura D.51: Cenario 61

Listagem D.52: Cenario 62

```
Given Fred has bought a microwave
And the microwave costs 100$
And the microwave was on 10% discount
When we refund the microwave
Then Fred should be refunded 90$
```

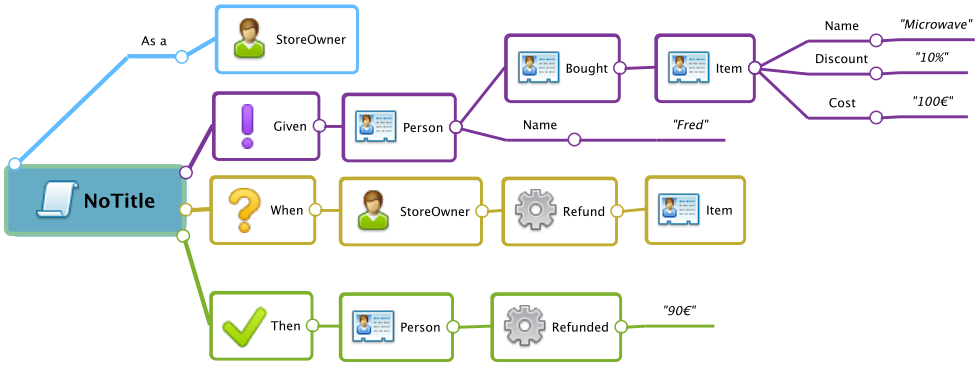


Figura D.52: Cenario 62

Listagem D.53: Cenario 63

```
Story: Adding content
    As a digital library user
    I want to add content to the library
    So that I can help improve the amount of documents

```

```
Scenario 1: Guest users can't add content
  Given I am at the digital library portal as a guest user
  When I try to add content
  Then I see 'Access Denied' error message
```

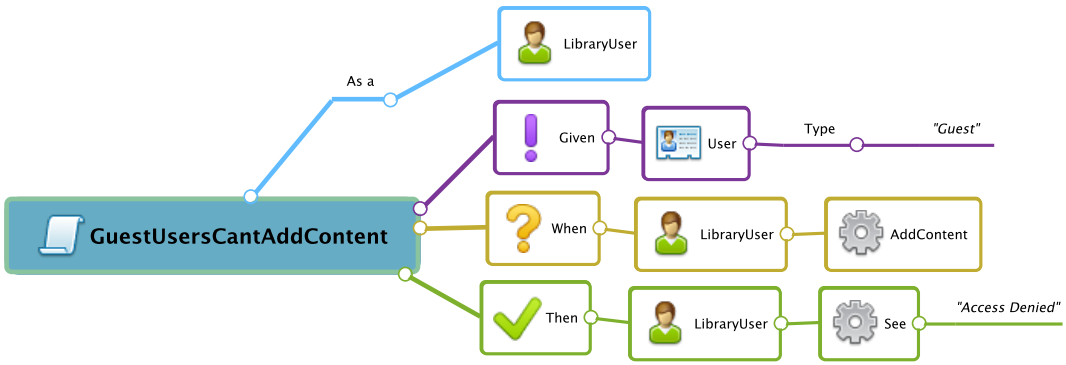


Figura D.53: Cenario 63